Foreword

This is the Second Workshop on Improving Non-English Web Searching (iNEWS'08) held in conjunction with the 17th ACM Conference on Information and Knowledge Management (CIKM 2008) in Napa Valley, California, on 30 October 2008. The First iNEWS'07 Workshop took place in Amsterdam (The Netherlands) in conjunction with the 30th Annual International ACM SIGIR Conference (SIGIR'07). The workshop aims at bringing together researchers interested in the issues surrounding non-English web searching.

Nowadays, over 60% of Internet users are non-English speakers and the number of non-English speaking Internet users is growing faster than the English speaking. Recent studies showed that non-English queries and unclassifiable queries have nearly tripled since 1997. Most search engines were originally engineered for English. These do not take into full account the specifics of non-English languages, such as, inflectional semantics nor diacritics or capitalization.

The main conclusion from the literature is that searching using non-English and non-Latin based queries results in lower retrieval success and requires additional user effort so as to achieve acceptable satisfaction levels. Furthermore, international search engines, like MSN Live, Google and Yahoo, are relatively weaker with monolingual non-English queries.

New tools and resources are needed to support researchers in non-English retrieval, new methodologies need to be proposed which will help the identification of problems in existing search engines and new teaching strategies should be formed aiding users to become more efficient in formulating their queries. Foremost, research in non-English web search should provide an incentive to search engines to improve the retrieval performance of their engines for non-English languages.

Taking into account these needs, the main objectives of this workshop are the proposal of techniques and the evaluation of tools which improve the effectiveness of the existing search engines. This way, the specific aims of the workshop have been to:

- Evaluate search engines in non-English queries and measure the additional user effort.
- Define methodologies for evaluating the effectiveness of search engines in non-English queries.
- Study the user query patterns in non-English Web retrieval.
- Identify the factors that influence utilization of search engines in a multicultural world.
- Propose extensions to the search engines to improve non-English Web retrieval.
- Propose teaching strategies for helping users improve their searching behavior.
- Identify how standard IR techniques (Indexing, Query representation, Query reformulation, etc) can be adapted in Web retrieval for non-English languages.
- Discuss the application of natural language processing techniques for non-English Web IR.

In response to our call, 27 papers were submitted. Each paper was reviewed by three expert referees in a blind review process. 10 full papers and 7 short papers were selected for oral or poster presentations and appear in these proceedings.

Finally, we wish to thank the CIKM'08 organizers, the program committee and our sponsor, the "Rede Galega de Procesamento da Linguaxe e Recuperacion de Informacion (Galician Network for Language Processing and Information Retrieval)," funded by the Xunta de Galicia government, for its support.

Fotis Lazarinis, Efthimis N. Efthimiadis, Jesus Vilares, John I. Tait

Napa Valley, California
October 30th, 2008

iNEWS'08 Workshop Organization List

Invited Workshop Chairs:	Fotis Lazarinis *(Technological Educational Institute of Mesolongli, Greece)*
	Efthimis N. Efthimiadis *(University of Washington, USA)*
Workshop Chairs:	Jesus Vilares *(University of A Coruna, Spain)*
	John I. Tait *(Information Retrieval Facility, Austria)*
Program Committee:	Mustafa Abusalah *(C.C.C. Ltd, Greece)*
	Miguel Alonso *(University of A Coruna, Spain)*
	Amit Bagga *(Comcast / StreamSage, USA)*
	Judit Bar-Ilan *(The Hebrew University of Jerusalem, Israel)*
	Richard Cai *(Microsoft Research, USA)*
	Raman Chandrasekar *(Microsoft Research, USA)*
	Keh-Jiann Chen *(Academia Sinica, Taiwan)*
	Kuang Hua Chen *(National Taiwan University, Taiwan)*
	Zheng Chen *(Microsoft Research Asia, China)*
	Jean-Pierre Chevallet *(Université Joseph Fourier, France)*
	Theodore Dalamagas *(National Technical University of Athens, Greece)*
	Francesca de Jong *(University of Twente, The Netherlands)*
	Arjen P. de Vries *(CWI, The Netherlands)*
	Carlos G. Figuerola *(University of Salamanca, Spain)*
	Alexander Gelbukh *(National Polytechnic Institute, Mexico)*
	Julio Gonzalo *(UNED, Spain)*
	Kalervo Jarvelin *(University of Tampere, Finland)*
	Gareth Jones *(Dublin City University, Ireland)*
	Ghassan Kanaan *(Yarmouk University, Jordan)*
	Dimitris Kanellopoulos *(Technological Educational Institute of Patras, Greece)*
	Sotos Kotsiantis *(University of Patras, Greece)*
	Paul McNamee *(Johns Hopkins University, USA)*
	Mandar Mitra *(Indian Statistical Institute, India)*
	Iadh Ounis *(University of Glasgow, UK)*
	Gabriel Pereira *(Universidade Nova de Lisboa, Portugal)*
	Stelios Piperidis *(Institute for Language and Speech Processing, Greece)*
	Vassilis Plachouras *(Yahoo! Research, Barcelona, Spain)*
	Hema Raghavan *(Yahoo! Research, USA)*
	Ian Ruthven *(University of Strathclyde, UK)*
	Yutaka Sasaki *(University of Manchester, UK)*
	Jacques Savoy *(University of Neuchatel, Switzerland)*
	Nasredine Semmar *(LIC2M/CEA-LIST, France)*
	Sofia Stamou *(University of Patras, Greece)*
	Anastasios Tombros *(Queen Mary University of London, UK)*
	Panayiotis Tsaparas *(Microsoft Research, USA)*
	Vasudeva Varma *(International Institute of Information Technology, India)*
	Manuel Vilares *(University of Vigo, Spain)*

Sponsors: SIGIR Special Interest Group on Information Retrieval — sigweb

Supporter: XUNTA DE GALICIA — "Rede Galega de Procesamento da Linguaxe e Recuperacion de Informacion (*Galician Network for Language Processing and Information Retrieval)*"

Efficient Multi-Word Expressions Extractor Using Suffix Arrays and Related Structures

José Aires
Universidade Nova de Lisboa
FCT/DI/CITI
aires@di.fct.unl.pt

Gabriel Lopes
Universidade Nova de Lisboa
FCT/DI/CITI
gpl@di.fct.unl.pt

Joaquim Silva
Universidade Nova de Lisboa
FCT/DI/CITI
jfs@di.fct.unl.pt

ABSTRACT
For Information Retrieval purposes, there is a need for regularly processing predictably dynamic and potentially huge corpora for extraction of contiguous Multi-Word Expressions (MWEs), in a way that should be computationally tractable. In this paper we'll be mainly exploring the use of Suffix Arrays, together with the *SCP* association measure and the *Smoothed LocalMaxs* algorithm. The choice of Suffix Arrays and the construction of auxiliary structures enabled a clear minimization of the time for extracting multi-word expressions, with linear complexity by the introduction of a limitation on the number of words. Despite the methodology being essentially of a statistical nature, we show how to handle hybrid extraction mechanisms.

Categories and Subject Descriptors
I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *text analysis*.

General Terms
Algorithms, Performance, Design, Experimentation, Languages.

Keywords
Multi-Word Expressions, Extraction, Suffix Arrays, Language Independent, Large Corpus.

1. INTRODUCTION
In the framework of European project ASTROLABIUM[1], our team should enable a Portal web page (traditionally designed) to support queries either in English or in Portuguese about every topic related to Mobility. This meant that the information about any Portuguese University and Research Center or about any subject matter related to scientific mobility was periodically crawled, fetched, separated by language (Silva et al, 2006) and full text indexed. To help users to easily find out the information they were looking for, fetched content of web pages (in English and in Portuguese) should be mined for extracting MWEs (Multi-Word Expressions). These expressions should then be made available to users, as topic hints, in order to help them to formulate their own information needs while making them aware of the content available. For this purpose (Information Retrieval), we needed to regularly process predictably dynamic and potentially huge corpora for language independent extraction of contiguous MWEs, in a way that should be computationally tractable (efficient and effective in terms of time and space). In this paper we'll mainly focus on our solution for computational tractability, by exploring the use of Suffix Arrays, together with an association measure (*SCP*) and an algorithm (*Smoothed LocalMaxs*) which improves the *LocalMaxs* algorithm presented in (Silva et al, 1999). The choice of Suffix Arrays was due to intrinsic properties of this data structure for full text indexing and for efficiently counting the frequency of occurrence of every sub-string in a text collection (Yamamoto and Church, 2001). The choice of the *SCP* association measure working together with the *Smoothed LocalMaxs* algorithm was due to: the comparatively high precision they enable, when compared with various other methods (Silva et al, 1999); their language independent character; and their capability to deal naturally with MWEs made of two or more words. It must be stressed that both the *SCP* and the *Smoothed LocalMaxs* are extremely demanding in terms of calculations. But the choice we did of Suffix Arrays, together with auxiliary data structures and adequate algorithms, as will be shown along this paper, enabled a clear minimization of the time for extracting MWE's, with linear complexity.

[1] ASTROLBIUM was a Specific Support Action, under **contract MOBI-CT-2003-003344.**

2. BACKGROUND AND MOTIVATION
MWEs such as compound nouns (*British minister of foreign affairs*, *World Trade Center*), compound verbs (*to take into account, kick the bucket*), compound prepositions or prepositional locutions (*as a result of*), compound conjunctions (*in order to*), frozen phrases (*raining cats and dogs*), adverbial locutions (*from time to time*), compound determiners (*a huge amount of, a set of*), and many others, do actually appear in real texts and are rarely represented in NLP lexicons and in thesauri automatically constructed (Gamallo et al, 2005), preventing parsers from being more effective and efficient. MWEs availability can help users to refine their queries, enhancing precision, recall and interaction friendliness between users and search engines. They can also be used in parallel text alignment or as guides for extracting MWE translations from aligned parallel corpora (McNamee and Mayfield, 2006).

MWEs extraction has been approached generally using different strategies: linguistic techniques (Blanck, 1998; Bourigault, 1996; Dagan, 1994), statistical techniques (Chengxiang, 1997; Church et al, 1990; Shimohata, 1997; Smadja, 1993; Silva et al, 1999) and a combination of both (Bourigault et al., 2001; Jacquemin & Bourigault, 2003).

Language-dependent techniques based on linguistics exploit the morpho-syntactic structure of MWEs and require specific language processing operations, namely Part-of-speech (POS) tagging, and some syntactic parsing.

Language independent procedures based on statistical techniques determine the degree of cohesiveness between constituents of possible MWEs using several association measures, sort those expressions by decreasing order of their value of cohesiveness and must empirically find out a threshold value that enables high precision/recall selection of MWEs. One of the problems they face is the choice of that threshold that depends on the extension of the corpus processed and on many other features (Smadja, 1993). Another problem that those approaches had not solved was related to their inability for extracting MWEs with more than 2 words. Both problems were solved by Silva et al (1999).

Hybrid approaches, combining linguistic and statistical techniques, are also applied, mainly in two manners: statistical proceeding is used to filter the term candidates obtained through linguistic techniques (Church et al 1991; Smadja, 1991; Seretan and Wehrli, 2006), and, vice versa, some linguistic filters are exploited after statistical processing (Daille, 1995; Enguehard, 1993; Justeson and Katz, 1995), to extract the statistically significant word combinations that match some given syntactic patterns.

So, we might say that the extraction of MWEs exploits rather well-established techniques that seem difficult to improve significantly, except in what concerns computational efficiency when dealing with large volumes of text. For this purpose we needed data structures powerful enough for efficient string frequency counting, for full text indexing, for efficient string matching and retrieval. Suffix Arrays adequately support these and other needs. Yamamoto and Church (2001) is among the users of this data structure for Text Mining purposes. Relatively recent results (reported by Abouelhoda, 2004), related to the construction and use of Compressed and Succinct Suffix Arrays led us to prefer Suffix Arrays to Suffix Trees. Our choice was made on the basis of favorable space and time constraints.

3. *SMOOTHED LOCALMAXS* ALGORITHM

To decide if a word *n*-gram *W* is a multi-word expression, we use the *Smoothed LocalMaxs* algorithm described bellow

$\forall x \in \Omega_{n-1} \wedge \forall y \in \Omega_{n+1}$ W is a MWU if

$$\left(length\ (W) = 2 \wedge g(W) > y\right) \vee \left(length\ (W) > 2 \wedge g(W) > \frac{x+y}{2}\right)$$

where function *g*(.) represents a general association measure which we'll instantiate with *SCP*(.) (see next section), Ω_{n-1} is the set of values determined for the association measure of all contiguous n–1 grams contained in *W* and Ω_{n+1} is the set of values determined for the association measure of all contiguous n+1 grams containing *W*. This algorithm is more flexible than the original *LocalMaxs* algorithm (Silva et al, 1999). Instead of requiring that the association measure for the selected MWE is a strict local maximum, it requires it to be higher than the medium value of *x* (the highest association measure of the two sub-expressions with *n*–1 words) and *y* (the highest association measure of all super-expressions of *n*+1 words), according to the formula. This way, if "*fleet capacity*" is extracted "*fleet capacity policy*" and "*Community fleet capacity policy*" may also be extracted. According to original *LocalMaxs* algorithm (Silva et al, 1999), if an *n*-gram *W* was selected as a MWE, neither a sub-expression of *W,* with *n*–1 words nor a super-expression of *W,* with *n*+1 words, could be selected as MWEs, meaning that, if "*fleet capacity policy*" was selected, neither "*Community fleet capacity policy*" nor "fleet *capacity*" would too. As a consequence recall improved while precision slightly decreased. Moreover, the selection of MWEs does not depend on any empirically determined threshold. An expression with very low value of *g*(.) can be selected if it's a smoothed local maximum, and a word *n*-gram with high value of *g*(.) may not be selected if it's not a smoothed local maximum. These results don't depend on the instantiation of *g*(.).

As we are dealing with contiguous *n*-grams, with *n* equal or larger than two, having no gaps, there are only two sub-expressions of *W* with *n*–1 words. The number of expressions with *n*+1 words containing *W* depends on the number of different words occurring before and after *W*. This means that for selecting an expression *W* with *n* words, the algorithm requires an access to values of *g*(.) for the two sub-expressions of *W* having *n*–1 words and for all super-expressions of *W* with *n*+1 words.

We might process everything on a need-to basis, but that leads to many recalculations that can and shall be avoided. For this purpose, we pre-calculate all the association measure values and only then determine the maximum ones, without calculating the sets explicitly, as described in section 6.2.

To help identify these two quantities, we shall call "*x*" as the maximum inner value and "*y*" as the maximum outer value.

4. *SCP* ASSOCIATION MEASURE

As mentioned in the previous section, we need an association measure to assess a term's relevance. The *SCP* (from *Symmetric Conditional Probability*) (Silva et al 1999) is an association measure that, for a given bi-gram *W*, composed of two words, $[w_1w_2]$, is defined as follows:

$$SCP(W) = SCP([w_1w_2]) = p(w_1 \mid w_2) \cdot p(w_2 \mid w_1) = \frac{p(w_1w_2)^2}{p(w_1) \cdot p(w_2)}$$

For cases in which *W* has more than two words ($W=[w_1,\ldots,w_n]$, with $n>2$), as we don't know in advance where to break the *n*-gram, we assume that it may be broken at any interior point and viewed as a pseudo-bigram made of two sequential grams, an *i*-gram and an *n-i*-gram. To be fair with these pseudo-bigams the denominator of *SCP* is determined as an average value of the products of probabilities of all sequential *i*-grams and *n-i*-grams, with *i* varying from 1 to *n*–1, as shown bellow

$$Avp = \frac{1}{n-1}\sum_{i=1}^{n-1} p(w_1 \ldots w_i) \cdot p(w_{i+1} \ldots w_n)$$

And a fair *SCP*, *SCP_f*, is determined using the next equation

$$SCP\_f([w_1 \ldots w_n]) = \frac{p(w_1 \ldots w_n)^2}{Avp}$$

4.1 *SCP* as a Function of Frequencies

The probability of a term with *n* words W_n, within a text with *N* words, is $p(W) = \frac{tf(W)}{N-n+1}$, where $tf(W_n)$ is the term frequency of

W_n and N–n+1 is the number of terms with n words in the text. Since N is much larger than n, N nicely approximates N–n+1, and so $p(W) \cong \frac{tf(W)}{N}$. This way, instead of the word n-grams' probabilities, we can just use their frequencies

$$SCP(w_1w_2) = \frac{p(w_1w_2)}{p(w_1)\cdot p(w_2)} = \frac{\left(\frac{tf(w_1w_2)}{N-1}\right)^2}{\frac{tf(w_1)}{N}\cdot\frac{tf(w_2)}{N}} \cong \frac{tf(w_1w_2)^2}{tf(w_1)\cdot tf(w_2)}$$

The same change can be applied to the *SCP_f* function.

$$Avp = \frac{1}{n-1}\sum_{i=1}^{n-1}\frac{tf(w_1..w_i)}{N-i+1}\cdot\frac{tf(w_{i+1}..w_n)}{N-(n-i)+1} \cong \frac{1}{n-1}\sum_{i=1}^{n-1}\frac{tf(w_1..w_i)}{N}\cdot\frac{tf(w_{i+1}..w_n)}{N} =$$

$$= \frac{1}{N^2}\cdot\frac{1}{n-1}\sum_{i=1}^{n-1} tf(w_1...w_i)\cdot tf(w_{i+1}...w_n) = \frac{1}{N^2}Avp\_tf$$

with

$$Avp\_tf = \frac{1}{n-1}\sum_{i=1}^{n-1} tf(w_1...w_i)\cdot tf(w_{i+1}...w_n)$$

resulting in

$$SCP\_tf([w_1...w_n]) \cong \frac{\left(\frac{tf(w_1...w_n)}{N}\right)^2}{\frac{1}{N^2}Avp\_tf} = \frac{tf(w_1...w_n)^2}{Avp\_tf}$$

This change to use frequencies instead of probabilities is very convenient due to the structures we'll be using, while avoiding intermediate calculations that would certainly lead to rounding problems, as word n-gram probabilities are very small.

5. SUFFIX ARRAYS AND ASSOCIATED STRUCTURES

5.1 Preprocessing Considerations

Before proceeding to technical explanation of used data structures, we must make clear three distinct working aspects. First, as we'll mainly be focusing on Western languages, we take words (or tokens) as our basic units, but as it will be seen later this methodology can be applied to Eastern languages, where characters will then be the basic unit. This is because if we were using characters on Western languages, applying the *SCP* formula would mean that to find expression "*European Community*" we would be dealing with 18 characters (counting the space), and 19 characters when we include the *LocalMaxs* algorithm, checking for the set of terms containing it, risking to obtain expressions like "*pean Comm*" or "*n Com*", which were not interesting for our purposes; we know we only need to focus on two words ("*European*" and "*Community*") for the *SCP* and three for the *LocalMaxs*. Second, for constructing necessary data structures for representing input text we opted to use their lower-case representations. Finally, in order to simplify the process, every distinguishable element (word, number, punctuation symbol, …) is separated by only one blank space and, to avoid testing for limit cases, this blank space is also included in the beginning and in the end of the text. Any kind of combination of several separator symbols like tabs, carriage returns or line feeds will be replaced by only one blank space. This adjustment enables us to just focus on the words.

5.2 Suffix Array

Given a text T of size N words, let $T[i]$ denote the suffix of T spanning from offset i until the end of the text. The Suffix Array SA of text T is an array of offsets sorted according to the lexicographic order of the corresponding suffixes. The resulting array will be ordered by suffixes, for which we have $T[SA[0]]<T[SA[1]]<…<T[SA[N]]$. The Suffix Array for text T="to be or not to be" is shown in Table 1.

Table 1: Suffix Array

i	*SA*[*i*]	*T*[*SA*[*i*]]
0	16	be
1	3	be or not to be
2	9	not to be
3	6	or not to be
4	13	to be
5	0	to be or not to be

5.3 LCP Array

The LCP Array keeps the length of the longest common prefix ($LCP[i]$) of every two adjacent Suffix Array entries ($SA[i]$ and $SA[i–1]$). It has N+1 entries, since a first and a last entry with zero are added. This structure is used to help create the Suffix Class Array (see next subsection). The LCP Array for the Suffix Array represented in Table 1 is shown in Table 2.

Table 2: LCP Array

i	*LCP*[*i*]	*T*[*SA*[*i*]]	*T*[*SA*[*i*–1]]
0	0	be	(out of bounds)
1	2	be or not to be	be
2	0	not to be	be or not to be
3	0	or not to be	not to be
4	0	to be	or not to be
5	5	to be or not to be	to be
6	0	(out of bounds)	to be or not to be

Every entry has the LCP compared to the previous entry, except the first and last, which have elements out of bounds. These two entries (with value 0) are added to avoid checking for limit cases in the Suffix Class Array construction.

5.4 Suffix Class Array

The Suffix Class Array represents every term present in the text with every Suffix Class representing a set of adjacent Suffix Array entries, sharing a common prefix and having the same frequency, condensing in it every instance of a given term. This will be useful to keep information about every term present in the corpus, getting their frequency and avoiding dealing with repetitions. Also, as we'll see later, its locality can be explored to improve the process's performance. The Suffix Class Array for the Suffix Array represented in Table 1, using the corresponding LCP Array represented in table 2, is shown in Table 3.

Table 3: Suffix Class Array

i	LBL	SIL	*SA* range	tf	text
0	0	2	[0, 1]	2	be
1	2	15	[1, 1]	1	be or not to be
2	0	9	[2, 2]	1	not to be
3	0	12	[3, 3]	1	or not to be
4	0	5	[4, 5]	2	to be
5	5	18	[5, 5]	1	to be or not to be

Where the fields have the following meaning:

- LBL: Longest Boundary LCP. Only whole-word expressions with length beyond this number belong to the

Suffix Class. For instance, index 1 has expressions «be or», «be or not», …, «be or not to be» (check Table 4); «be» doesn't belong to index 1 since its length is not beyond 2 – it belongs to index 0.

- SIL: Shortest Internal LCP. Can be seen as the limit length for whole-word expressions of the Suffix Class.
- *SA* range: The range of Suffix Array indices of entries having prefixes belonging to the Suffix Class.
- tf: the term frequency.

Its construction requires the LCP Array and uses a stack to explore nested characteristics of many of the Suffix Classes.

Unlike what might be expected without this structure, to get the term frequencies of every expression we don't need to process all possible expressions, which would be $\frac{N}{2}(N+1)$ for a text of N words. The Suffix Class Array, with a lot less elements (up to $2N-1$), makes the term frequency information available for every expression, making the task more tractable.

5.5 Term Array

The purpose of the Term Array is to partition every Suffix Class, individually representing all their terms. The Term Array for the Suffix Class Array represented in Table 3 is shown in Table 4.

Table 4: Term Array

i	*SC* index	term
0	0	be
1	1	be or
2	1	be or not
3	1	be or not to
4	1	be or not to be
5	2	not
6	2	not to
7	2	not to be
8	3	or
9	3	or not
10	3	or not to
11	3	or not to be
12	4	to
13	4	to be
14	5	to be or
15	5	to be or not
16	5	to be or not to
17	5	to be or not to be

The *SC* index is the Suffix Class index to where each term belongs. This shows that term «be» belongs to Suffix Class 0 (check Table 3), «be or», «be or not», «be or not to» and «be or not to be» belong to Suffix Class 1, and so on. These terms will include some associated data, namely the *SCP* and the maximum inner and outer values for the *Smoothed LocalMaxs* algorithm.

This partition is necessary since the *SCP* is not shared by terms of the same Suffix Class. To show this, Table 5 presents the *SCP* values of the *n*-grams (n>=2) of Suffix Class 3 (Table 3).

Table 5: *SCP* values

term	*SCP*
or not	1
or not to	0.667
or not to be	0.6

This is a consequence of having suffix terms with a higher frequency, affecting the *SCP* of the corresponding Suffix Class terms. Yamamoto et al (2001) already warned for the possibility of elements of the same Suffix Class not sharing some statistic properties, unlike what happens with the term frequency.

5.6 Changes to the Structures

A few changes have been introduced in order to adapt the above structures to the word-based concept, since they were initially conceived as being character-based. These changes affect the Suffix Array, the LCP Array and the Suffix Class Array. The Term Array is a structure created by us having the word-base concern since its inception. Besides, there was also the need to include adjustments to exclude expressions having more than a limit number of words or including unwanted symbols, such as characters not referring to words (punctuation or numbers).

Even though our unit is the word, we keep offsets relative to the character instead because they offer a more efficient way to get to the expressions in the original corpus, as well as to deal with lengths. So, instead of having the n^{th} word, we have the m^{th} character, which is the offset for the n^{th} word; the number of words in a given expression is calculated while it's being processed. Besides, the comparisons are character-based since it makes easier the access through the offsets.

5.6.1 Suffix Array

The Suffix Array is character-based built. But since we were only interested in having words as units, we took the character-based implementation and turned it into word-based, not requiring any more adjustments to it. Since we were also interested in keeping the sorting method (Larsson, 1999) without introducing profound changes, we simply created a character-based Suffix Array and discarded the entries that didn't refer to words, which consisted in simply taking the entries starting with a blank and then adding them by one, since we want the offset of the word and not the offset of the blank preceding it. We didn't invest some effort on creating an alternative for building the Suffix Array solely on words since to establish an order between them we need to access their characters anyway. A possibility could be to map every word to an id, but this change would leave the algorithm unaffected, would require an extra structure and then we would need to change them back to the originals to check results. Anyway, we were more interested in the gains of not dealing with all the characters, significantly reducing our search space and not using an extra structure.

5.6.2 LCP Array

We had to change the LCP Array creation criteria in order to not only reflect the same word-based changes to the Suffix Class Array, but also to discard expressions having unwanted symbols (those not referring to words: punctuation signs, brackets, …) in the middle or having more than a certain number of words. We needed the offsets (they are useful because of the expressions lengths – see below) so the comparisons are still character-based but on an all-or-nothing basis, meaning that either we have a whole word match or we don't have a match at all.

5.6.3 Suffix Class Array

Finally, we also had to adjust the Suffix Class Array, dealing once again with the character-based comparisons, but ensuring that only complete words would match. Another change was introduced in SIL, for which Yamamoto and Church (2001) use the value "infinity" to mean that the sequence is to be considered until the end of the text. Since we have other limitations (like the number of words) we use the actual term length. Besides, the original implementation of the Suffix Class Array creation resulted in a partial order of the Suffix Classes, where «be or not to be» would come before «be», in spite of both appearing before, «not to be», a consequence of using a stack in such process. We needed to change this to be able to avoid the prefix term lookup when we are processing the association measure, as will be shown in section 6.1. We simply order the entries in the end.

6. IMPLEMENTATION

To extract MWEs from a corpus, we need to efficiently process every expression using their *SCP* and the *Smoothed LocalMaxs* selection algorithm. Careful examination of both processes shows that our problem is mainly of a retrieval nature, since for each expression we need to retrieve information about a set of sub- and super-expressions. For instance, for processing the *SCP* of «be or not», we need the frequency of the expression itself and of «be», «or not», «be or» and «not»; and for selecting it, we need the *SCPs* of «be or», «or not», «to be or not» and «be or not to».

6.1 Calculating the Association Measure

Every expression in the corpus needs to have its *SCP* calculated in order to be used in the algorithm to determine if it's a MWE, so we need to know its frequency and the frequency of the expressions that compose it. These last ones are all the prefix expressions $w_1 \dots w_i$ and the suffix expressions $w_{i+1} \dots w_n$, with $i=\{1, \dots, n-1\}$ and n being the number of words in the expression. This means we need to know the frequencies of all expressions present in the text (with a maximum word length). As they'll be seldom used, we decided to pre-compute them all using the Suffix Class Array. Now, all we have to do is iterate through this array as a way to iterate through the Terms Array, checking every single one of them for their prefix and suffix expressions, to get their frequencies.

While traversing the Terms, each will be put in a stack to avoid the lookup for the prefix terms for the following terms to be processed. These stack terms are removed when their length is greater than the LBL from a Suffix Class being processed. When processing a new Term, the stack can be accessed in sequence, as in the i^{th} stack term will be the i^{th} prefix term of the Term being processed from a Suffix Class. This means we are using a stack that allows accessing its members through their offsets and not only its top element.

To clarify this idea, take the sample Suffix Classes in Table 6.

Table 6: Suffix Classes sample

text	LBL	SIL	tf
A	0	1	9
A A	1	2	1
A B	1	2	2
A B B	2	3	1
A B C	2	3	1
A C A	1	3	1
A D A	1	3	1

As mentioned before, the Suffix Classes are lexicographically ordered. Taking the previous table, we can illustrate what happens when we iterate through the Suffix Class Array. Table 7 shows the contents of the structure after traversing the Suffix Classes «A», «A A», «A B» and reaching «A B B».

Table 7: Contents when «A B B» is reached

index	0	1	2
term	A	A B	A B B
length	1	2	3
tf	9	2	1

Continuing the Suffix Class traversal, when «A B C» is reached, we can see that the last (right-most) entry «A B B» has length 3 (in Table 7), which is greater than the LBL from the «A B C» Suffix Class, so the «A B B» Term is removed and «A B C» is inserted, as shown in Table 8.

Table 8: Contents when «A B C» is reached

index	0	1	2
term	A	A B	A B C
length	1	2	3
tf	9	2	1

Then, when the «A C A» Suffix Class is reached, we can see that the last two entries in Table 8 («A B» with length 2 and «A B C» with length 3) were removed, since both their lengths were greater than the LBL of the «A C A» Suffix Class. In turn, the Terms «A C» and «A C A» were added to the structure (Table 9).

Table 9: Contents when «A C A» is reached

index	0	1	2
term	A	A C	A C A
length	1	2	3
tf	9	2	1

At this point, when processing «A C A», we already have the prefix terms «A» and «A C». This gives an idea of how things work to avoid the prefix term lookup.

However, for the suffix expressions, things aren't so simple, since they can be far apart from the original expression. We have to look for them in the whole set of Suffix Classes but we can do that through efficient binary searches.

After traversing all expressions, all the *SCPs* will have been calculated. Only then we may proceed to the phase of selecting MWEs using the *Smoothed LocalMaxs* algorithm.

6.2 Implementing MWEs Selection Algorithm

Looking at the *Smoothed LocalMaxs* algorithm description, it becomes evident that, for each expression, we need the information about its own *SCP* and the *SCPs* of its sub-expressions (obtained from the initial expression removing a limit word) and super-expressions (the ones from which we can obtain the initial expression by removing a limit word from them).

However, taking a closer look at the algorithm's description, we can see that having W_n and W_{n+1} can also be seen as having W_{n-1} and W_n, respectively. For instance, "be or not" belongs to the Ω_{n-1} set of "be or not to" and this last expression belongs to the Ω_{n+1} set of the first. This reflexive property is the basis for simplifying the calculation process.

To help understanding, let's say we have a term $W=w_1 \dots w_n$. From it we can easily obtain its sub-expressions $Ws=w_2 \dots w_n$ and $Wp=w_1 \dots w_{n-1}$, by removing extremity words w_1 and w_n, respectively. Ws and Wp belong to the Ω_{n-1} set of W (in fact, this set has no more members) and W belongs to the Ω_{n+1} set of Ws and Wp (frequently composed by many more members). This means that with this methodology we can immediately calculate the maximum inner value of *W*, but we need to process all the terms to guarantee the maximum outer values of *Ws* and *Wp* have also been obtained. Doing this for all terms will result in each of them having all their maximum inner and outer values determined.

This procedure is done only after the association measures have been calculated and, in the beginning of the procedure, the maximum inner and outer values of every expression are set to zero, since we are looking for maximums. Once again, this is done by traversing the Term Array through the Suffix Class Array to determine the greatest inner and outer association measures of all terms, exploring the mentioned reflexive nature of the relation.

As a consequence, since bi-grams will only have super-expressions and no sub-expressions, they don't need to be processed directly, since they'll be processed indirectly while processing the tri-grams. For this reason, we only have to access *n*-grams with three or more words.

To show how much we gain with this approach, we can see that for a given expression W_n we have only two expressions belonging to Ω_{n-1} (obtained by removing the extremity words), but we can have several expressions belonging to Ω_{n+1}, and their access will be very expensive (and unnecessary). This shows that we spare a lot of unnecessary calculations, especially with very frequent expressions, by avoiding the explicit search for super-expressions because instead of finding all super-expressions of a given expression we find the expressions to which the given expression is super in a two-in-one operation. When we finish the traversal, all the values will have been calculated.

In terms of efficiency, *Wp* will be close to *W*, since it's a prefix of *W*, so we can avoid looking for it by keeping a reference, but *Ws* will have to be obtained through a binary search because there is no guarantee to where it can be located. This is a similar problem to the one described for the association measure.

7. IMPROVEMENTS

There are two types of improvements applied to the process: eliminating candidates **before** the analysis and discarding terms **after** the analysis.

7.1 Before the Analysis

Before processing the terms, some restrictions have been introduced to limit the search space, having the reduction of the processing time as a goal. This reduction is mainly in terms of number of elements to process. However, this reduction had a tremendous impact on the final solution. This concern comes from the fact that a corpus can have a very large number of expressions and we should not process them all blindly, especially since we can eliminate a lot of candidates from the beginning, alleviating the process with unnecessary calculations.

7.1.1 Setting a Word Number Limit

Because of the very large number of candidates, and knowing that expressions with a considerable number of words aren't very interesting for our purposes, we limit the number of words composing an expression, heavily reducing the search space.

This limit is expressed by the maximum number of words we are admitting for a MWE, but since the *LocalMaxs* algorithm requires the set of Ω_{n+1} words, that limit is internally increased by one in order to process that auxiliary set, which won't be shown.

We have set the MWEs extraction to expressions having a maximum of seven words, since our experience indicates this is a good word count limit for the MWEs found, but it's merely empirical and can be changed to a larger number at the cost of more processing time (with progressively less significant improvements in the results) or to a smaller number with a smaller processing effort (with a significant loss of MWEs).

7.1.2 Eliminating Expressions Containing Unwanted Tokens

Considering that we aren't interested in expressions containing any kind of punctuation, we excluded these candidates from the beginning. However, we had to keep some limit elements.

For instance, we had to keep expressions like «, to be or not to be» (starting with a comma) and «to be or not to be .» (ending with a point) because they are necessary for the acceptance criteria of «to be or not to be», since they both belong to its super-expressions set, even though they won't be shown. Expressions like «, to be or not to be .» (starting and ending with some punctuation symbol) aren't necessary because they would be part of the super-expressions set of either «, to be or not to be» or «to be or not to be .», neither of which we are interested in.

This procedure may also be applied, in certain applications (e.g., IR), to filter out expressions starting with function words or ending with verb forms. This enables hybrid extraction of MWEs.

7.2 After the Analysis

After the first results have been obtained, we still needed a few improvements to discard a few elements that were not so interesting for the search reference purpose.

7.2.1 Eliminating Sub-Expressions

We realized we had many expressions that were sub-expressions of others having the same frequency, so we introduced a few changes to eliminate them. However, we only wanted to eliminate expressions that were sub-expressions of some expression that had already been considered as a MWE.

To illustrate what we mean by discarding the sub-expressions, let us take the example: "p*residente da república*" (*President of Republic*) and "p*residente da república portuguesa*" (*President of Portuguese Republic*). If both expressions have the same frequency (and both are considered as MWEs), it may be pointless for IR purposes, for example, to keep the former, since the latter is more specific and complete, occurring every time the former does. However, it should be noted that this wouldn't be the case if they had different frequencies, meaning we would have "p*residente da república*" occurring more often than "p*residente da república portuguesa*", resulting on the need to keep them both, since there would be some contexts in which the first would occur while the second wouldn't.

7.2.2 Elimination by Word Checking

For the purpose of IR, we still felt some more entries needed to be discarded according to their syntactic structure. So, we resorted to their elimination by checking against a few word syntactic patterns and against a set of words that have been shown to be very likely indicators that the expressions didn't have the characteristics we were looking for. This was only done for Portuguese, using a database with word morpho-syntactic information (verbs, nouns, adjectives, gender, number, …) and also checking for words occurring in their limits, as would be the case of function words ("*Comunidade*

*Europeia **e***" - "*European Community **and***", "***ou** a Câmara Municipal*" - "***or** the City Hall*", …), as well as checking for the occurrence of a set of words that have shown to be very good indicators of the poor quality of some entries (words like "*anterior*" – "*previous*", "*seguinte*" – "*following*", "*feito*" – "*done*" and "*verificado*" – "*checked*"). Application of these filters is optional as there are MWEs (as "*por exemplo*" – "*for instance*" or "*não obstante*" – "*in spite of*") that could be very useful for other purposes, like determining function multi-words.

7.2.3 Frequency Threshold

A frequency filter is also considered to discard MWEs occurring bellow a certain threshold. Such filter is to be set according to the size of the corpus being analyzed.

8. EXPERIMENTS AND EVALUATION

Our decision to use Suffix Arrays and their associated structures for efficiently extracting MWEs from huge corpora, especially when these corpora are highly dynamic and the set of MWEs must evolve along the time axis to keep valid the intrinsic occurrence relation between extracted MWEs and each specific corpus, has proven to be an adequate strategy for the entire process. We were able to process very large corpora in manageable time and obtaining good results. As an example, we were able to process a corpus of 350MB having around 70 million words in about 15 hours on a 3.2 GHz Intel Xeon, with 4GB of RAM and two SCSI disks of 200GB each, running Fedora Core 4.

Since the Suffix Class Array has the term frequency information about every possible term in a corpus in linear space, one could feel tempted to process other statistical features for every those terms. However, as it has been shown in Section 5.5, the *SCP* is not shared between members of the same Suffix Class, so we need to partition them, bringing us back to the quadratic space complexity. We overcome this by limiting the number of words in every term. So, in table 10 we show how much is gained by comparing the unlimited version (unlim.) to the limited one (lim.), where the maximum number of words is set to 7, simply to establish a limit. Unique terms means repetitions aren't considered.

Table 10

corpus size (bytes)	nr. of tokens	nr. of unique terms	nr. of possible terms	
corpus1 106248	21175	223998474	224179725	**unlim.**
		51840	70027	**lim.**
corpus2 980759	187430	361693823	17564908735	**unlim.**
		468293	797943	**lim.**

For corpus2, with less than 200K words, due to simply dealing with unlimited unique terms instead of working with all unlimited possible ones, there is a factor gain of almost 50. However, the most expressive gain is the one obtained by dealing with the limited unique terms instead of the unlimited possible ones, with a gain factor slightly greater than 37500.

To understand these gains, let's start by analyzing the total number of terms present in a corpus. Considering ec_n as being the number of existing terms with n words, we have $ec_1=N$, $ec_2=N-1$, …, $ec_{N-1}=2$, $ec_N=1$. So, generically, we have $ec_n=N-n+1$. The total number of existing terms consists in the sum of all these ec_n sequence terms, which is $1+2+\ldots+N-1+N=\frac{1+N}{2}N$, which in turn is the sum of the first N sequence terms of an arithmetic progression with a common difference of 1 and having 1 as the initial term. Now, considering we don't process 1-grams, the terms number will in fact be $\frac{N}{2}(N-1)$. Proceeding with our analysis, with the limit introduction we need to sum $ec_2+ec_3+\ldots+ec_{L-1}+ec_L$, being $L-1$ sequence terms, totaling $\frac{2N-L}{2}(L-1)$ terms, obtaining the previous total if $L=N$.

In fact we need the sum $ec_2+ec_3+\ldots+ec_{L-1}+ec_L+ec_{L+1}$, since we need to process W_{L+1} for the algorithm, in order to be able to classify a given W_L as a MWE. So the number of terms will be $\frac{2N-L-1}{2}L$, with L lower than N (usually much lower).

For the *SCP* of W_n, we need to obtain information about $2n-1$ different terms, since we have $n-1$ prefix terms, $n-1$ suffix terms, and the original term itself, making $n-1+n-1+1=2n-1$.

Now, considering nop_n as the number of operations needed to process all terms with n words, we have $nop_n=(2n-1)(N-n+1)$, with $1<n<=L+1$, meaning that, with the limitation introduction, we are not only reducing the number of processed terms, but discarding the ones that require the most processing, even though the greater the n, the lesser the terms with n words.

In fact, since we only deal with unique terms, the number of processed terms is $\frac{2N-L-1}{2}L+N_{SC}-\sum_{i=1}^{N_{SC}}tf(SC_i)$, where SC_i is the i^{th} member of the Suffix Class Array, N_{SC} is the number of Suffix Classes, and $tf(x)$ is the term frequency of x. If the term frequencies are all one, we get the previous complexity.

As it has been shown in Silva et al (1999b), *SCP* together with the filtering algorithm produced slightly better results than *Chi-squared* and *Specific Mutual Information*, when these association measures were used with the same kind of filtering algorithm. *Dice* and *Log-Like* produced substantially worse results. Complexity of using either *Chi-squared* or *Specific Mutual Information* is approximately the same as for *SCP* and *Smoothed LocalMaxs* algorithm, so we did not experiment these ones.

The main advantage of this processing approach is the way the information is organized, which allows us to process the *SCP* and the *Smoothed LocalMaxs* in a very efficient way, in spite of their complexity. The results obtained with the changes introduced in the implementation described in this paper were evaluated for the scope of the ASTROLABIUM project mentioned in Section 1, where we needed good quality nominative terms to be used as topic search suggestions, to which the changes mentioned in Subsection 7.2.2 had a major contribution.

Some examples of the MWEs obtained are: "*academia das ciências de lisboa*", "*aceitação da candidatura*", "*acessibilidade dos sítios da administração pública*", "*biologia celular e molecular*", "*ciclo básico do curso de medicina*", "*cidade universitária*", "*directivas para a acessibilidade*", "*director do instituto de estudos estratégicos*", "*ensino de línguas*", "*história das artes visuais*", "*leis e regulamentos académicos*", "*planos de prevenção e emergência em museus*", "*política externa nacional*", "*tipos de representações cartográficas*" and "*área departamental de ciências humanas e sociais*". These are just a sample from a total of 4622 accepted entries, making an acceptance ratio of 60% from the total candidates, extracted from a small corpus in Portuguese.

9. FUTURE WORK

In what concerns the extraction of MWES in Oriental languages, as we will deal with a limited number of characters, the same already appointed advantages will be achieved. However, we cannot resort to the same strategies to reduce the search space since there is not a clear notion of word or use of punctuation.

Our tokenizer may also be improved and avoid the concatenation of titles with each other and with paragraphs. It may also be adapted to specific languages and deal with initials or terms containing punctuation symbols such as "*Sr.*" – "*Mr.*" or "*E.U.A.*" – "*U.S.A.*", where those symbols should not be separated.

Despite the statistical nature of our approach, as it has been shown, it can handle pretty well with hybrid extraction methods with linguistic filters acting both prior and after statistical processing or even enabling the work on POS-tagged corpora the same way we work with raw text.

10. REFERENCES

[1] M. I. Abouelhoda, S. Kurtz and E. Ohlebush, 2004. Replacing Suffix Trees with Enhanced Suffix Arrays. Journal of Discrete Algorithms 2: 53--86, Elsevier.

[2] I. Blanck, 1998, Computer-Aided Analysis of Multilingual Patent Documentation. Proceedings of the First LREC, pp. 765-771.

[3] D. Bourigault, 1996, Lexter, a Natural Language Processing Tool for Terminology Extraction. Proceedings of the 7th EURALEX International Congress.

[4] D. Bourigault, C. Jacquemin, and M.-C. L'Homme 2001. Recent Advances in Computational Terminology, Natural Language Processing, 2(2)328-332, John Benjamins.

[5] Z. Chengxiang, 1997, Exploiting Context to Identify Lexical Atoms: a Statistical view of Linguistic Context. cmp-lg-9701001, 2 January 1997.

[6] K. Church et al, 1990, Word Association Norms Mutual Information and Lexicography. Computational Linguistics 16(1) 23-29.

[7] I. Dagan, 1994, Termight: Identifying and Translating Technical Terminology. Proceedings of the 4th Conference on Natural Language Processing, ACL.

[8] B. Daille, 1996. Study and Implementation of Combined Techniques for the Automatic Extraction of Terminology. In J. Klavans and P. Resnik, editors, The Balancing Act Combining Symbolic and Statistical Approaches to Language. pp. 49-66. Cambridge, Massachusetts: MIT Press.

[9] C. Enguehard, 1993. Acquisition de Terminologie a partir de Gros Corpus. Informatique & Langue Naturelle, ILN'93, pp. 383-394.

[10] P. Gamallo, A. Agustini and G. P. Lopes 2005. 'Clustering Positions with Similar Requirements. Computational Linguistics. 31(1): 107-145. MIT Press.

[11] C. Jacquemin and D. Bourigault 2003. Term Extraction and Automatic Indexing, Chapter 19, in R. Mitkov, editor, Handbook of Computational Linguistics Oxford University Press, Oxford.

[12] J. S. Justeson and S. M. Katz 1995. Technical Terminology: Some Linguistic Properties and an Algorithm for Identification in Text, Natural Language Engineering. 1(1):9-27.

[13] N. Larsson and K. Sadakane, 1999. Faster suffix sorting. Technical Report LU-CS-TR:99-214. Department of Computer Science, Lund University, Lund, Sweden.

[14] U. Manber and G. Myers. 1990. Suffix arrays: A new method for on-line string searches. In *Proceedings of The First Annual ACM-SIAM Symposium on Discrete Algorithms,* pages 319-327.

[15] P. McNamee and J. Mayfield, 2006. Translation of Multiword Expressions Using Parallel Suffix Arrays. Proceedings of the 7th Conference of the Association for Machine Translation in the Americas, pp. 100-109, Cambridge August 2006, AMTA

[16] V. Seretan and E. Wehrli, 2006. Accurate Collocation Extraction Using a Multilingual Parser. Proceedings of the 21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of ACL, Sydney, July 2006. pp.953-960

[17] S. Shimohata, 1997, Retrieving collocations by co-occurrences and Word Order Constraints. Proceedings of ACL-EACL. 476-481.

[18] J.F.Silva, G.Dias, S.Guilloré, J.G.P.Lopes. 1999. "Using LocalMaxs Algorithm for the Extraction of Contiguous and Non-contiguous Multiword Lexical Units". In P. Barahona, editor, *Progress in Artificial Intelligence: 9th Portuguese Conference on AI, EPIA'99, Évora Portugal September 1999, Proceedings*. LNAI series, Springer-Verlag, Vol. 1695, p. 113-132.

[19] J.F. Silva, and J.G.P.Lopes. 1999b. "A Local Maxima method and a Fair Dispersion Normalization for extracting multi-word units from corpora". In *Proceedings of the Sixth Meeting on Mathematics of Language (MOL6), Orlando, Florida July 23-25, 1999*. pp. 369-381

[20] J.F.Silva, and J.G.P. Lopes. 2006. 'Identification of Document Language is not yet a completely solved problem". In L. A. Zadeh and S. Grossberg, editors, *Proceedings of International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA), Sidney, Australia, 28 November to 1 December*. IEEE. 2006.

[21] F. Smadja, 1993, Retrieving collocations from Text: STRACT. Computational Linguistics 19(1), pp. 143-177.

[22] A. Voutilainen. 1993. NPtool. A detector of English noun phrases, Proceedings of the Workshop on Very Large Corpora, Columbus, Ohio.

[23] M. Yamamoto and K. Church, 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. Computational Linguistics 27(1): 1 – 30. MIT Press Cambridge, MA, USA

Towards an Error-Free Arabic Stemming

Eiman Tamah Al-Shammari
George Mason University
Dept. of Computer Science
4400 University Drive
Fairfax, VA 22030
Eiman.tamah@gmail.com

Jessica Lin , Ph.D
George Mason University
Dept. of Computer Science
4400 University Drive
Fairfax, VA 22030
jessica@cs.gmu.edu

ABSTRACT

Stemming is a computational process for reducing words to their roots (or stems). It can be classified as a recall-enhancing or precision-enhancing component.

Existing Arabic stemmers suffer from high stemming error-rates. Arabic stemmers blindly stem all the words and perform poorly especially with compound words, nouns and foreign Arabized words.

The Educated Text Stemmer (ETS) is presented in this paper. ETS is a dictionary free, simple, and highly effective Arabic stemming algorithm that can reduce stemming errors in addition to decreasing computational time and data storage.

The novelty of the work arises from the use of neglected Arabic stop-words. These stop-words can be highly important and can provide a significant improvement to processing Arabic documents.

The ETS stemmer is evaluated by comparison with output from human generated stemming and the stemming weight technique.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing – *Indexing methods, Linguistic processing*.

General Terms

Algorithms, Documentation, Experimentation, Human Factors, Languages, Standardization.

Keywords

Arabic, Lemmatization, Stemming, Text Mining, Tokenization.

1. INTRODUCTION

Stemmers are basic elements in query systems, indexing, web search engines and information retrieval systems (IRS). Stemming offers the benefits of minimizing storage requirements by eliminating redundant terms, as well as increasing matching probability for document comparison and unifying vocabulary [1].

Unfortunately, stemming can cause errors in the form of over-stemming, mis-stemming and under-stemming. These errors decrease the effectiveness of stemming algorithms [2] however reducing one type of errors can lead to an increase of the other [3].

Over-stemming occurs when two words with different stems are stemmed to the same root. An over-stemming example is when the word "probe" and "probable" are merged together after stemming.

Under-stemming occurs when two words that should be stemmed to the same root are not, for example, when the stemmer fails to conflate the words "adhere" and the word "adhesion" to the same root.

Mis-stemming is defined as "taking off what looks like an ending, but is really part of the stem [4] for example, stemming the word "red to "r" or the word "reply to "rep".

The challenges associated with stemming are even more pronounced in Arabic. Arabic is one of the most complex languages, in both its spoken and written forms. However, it is also one of the most common languages in the world. The Arabic language exhibits a very complicated morphological structure.

This paper presents a stemming algorithm that relies on Arabic language morphology and Arabic language syntax. The stemming algorithm automatically identifies nouns and verbs without the need for a dictionary. Nouns and verbs are stored in a separate dictionary. Automated addition to the syntactic knowledge and construction of corpus-based dictionaries reduce both stemming errors and stemming cost.

The remainder of this paper is organized as follows:

Section II includes a brief review of Arabic language morphology and discusses previous Arabic language stemming processes. Section III introduces the proposed methodology followed by a description of the stemming algorithm in section IV. Section V presents the evaluation criteria and experimental results. A conclusion and discussion of future work can be found in section VI.

2. BACKGROUND AND RELATED WORK

Arabic language is a semantic language with a composite morphology. Arabic words are categorized as particles, nouns, or verbs [5].

Unlike most western languages, Arabic script writing orientation is from right to left. There are 28 characters in Arabic. The characters are connected and do not start with capital letter as in English. Figure 1 below shows a list of Arabic characters. Furthermore, most of the characters differ in shape based in their position in the sentence and adjunct letters. Figure 2 below demonstrates some of the Arabic characters changes in shape.

خ	ح	ج	ث	ت	ب	ا
ص	ش	س	ز	ر	ذ	د
ق	ف	غ	ع	ظ	ط	ض
ي	و	ه	ن	م	ل	ك

Figure 1: Arabic characters (letters)

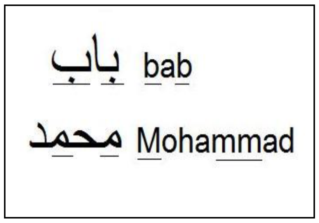


Figure 2: Arabic characters shape differ as their position in the word change

In Arabic, proper nouns do not start with capital letter as in English, which makes one particularly challenging task for machines, recognizing and extracting proper nouns from Arabic texts.

Furthermore, in English, words are formed by attaching prefixes and suffixes to either or both sides of the root. For example the word <u>*Untouchables*</u> is formed as follows

Un	*touch*	*able*	*s*
Prefix	Root	First Suffix	Second Suffix

In Arabic, additions to the root can be within the root (not only on the word sides) which is called an infix. This causes a serious issue in stemming Arabic documents because it is hard to differentiate between root characters (letters) and affix letters. For example, for the root "drink" **شرب** in Arabic, adding the infix letter "ا" (circulated in Figure 3) formed a different word: **شارب** "drinker".

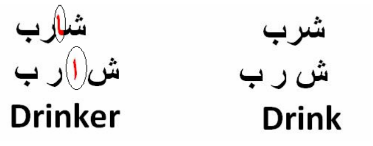


Figure 3. Arabic infix example

Table 1 displays an example of the Arabic Word = الشارب (drinker) and its stems with the common prefixes and suffixes.

Table 1. Arabic Example

Prefixes + Stem (Root + Pattern) + Suffixes		
Root	شرب	drink
Prefixes	ال	the
Stem	شارب	drinker
Suffixes	ين OR ان	dual
Suffixes	ون	plural
Suffixes	ة	feminine
الشاربان	the drinkers (dual)	
الشاربين	the drinkers (plural)	
الشارب	the drinker (masculine)	
الشاربه	the drinker (feminine)	

Suffixes, prefixes and infixes are categorized based on their uses. Similar to other Western languages, there are specific suffixes to convert the word from the singular form to the plural form and others to convert from masculine to feminine.

Due to its complicated morphological structure, Arabic requires a different stemming process from other languages.

Automatic Arabic stemming proved to be an effective technique for text processing for small collections [6-8] and large collections [9,10] of documents. Xu et al. [11] showed that spelling normalization combined with the use of tri-grams and stemming could significantly improve the accuracy of Arabic text processing by 40%. Additionally, in Al-Shammari et.al [6] it was proven that stemming can improve text clustering.

Stemming Arabic documents was performed manually prior to TREC (Text Retrieval Conference) and only applied on small corpora. Later, many researchers both native and non-native Arabic speakers created a considerable amount of Arabic stemming Algorithms.

Based on the required level of analysis, Arabic stemmers are categorized as either root-based [12, 13] or stem-based [9, 10, 14] .

In Arabic, the root is the original form of the word before any transformation process [15]. However, a stem is a morpheme or a set of concatenated morphemes that can accept an affix [16].

A superior root-based stemmer is the Khoja's stemmer[14], presented by Khoja and Garside[12] . The Khoja algorithm removes suffixes, infixes and prefixes and uses pattern matching to extract the roots. The algorithm suffered from problems especially with names and nouns.

A possible solution for this problem is to add a lookup dictionary to check the nouns, roots and names. Although this solution seems straightforward and easy, this process is computationally expensive. Al-Fedaghi and Al-Anzi [17] estimated that there are around 10,000 independent roots. Each root word can have prefixes, suffixes, infixes, and regular and irregular tenses.

On the other hand, there have been several proposed Arabic stem-based (light) algorithms [9, 10, 14, 18-20]. The prominent Arabic light stemmer is Aljlayl [14, 18] light stemmer. Light stemming does not deal with patterns or infixes; it is simply the process of stripping off prefixes and/or suffixes. Unfortunately, the unguided removal of a fixed set of prefixes and suffixes causes many stemming errors especially where it is hard to distinguish between an extra letter and a root letter.

Although light stemmers produce fewer errors than aggressive root-based stemmers; in contrast, aggressive stemmers reduce the size of the corpus significantly. Paice [3,21,22] proved that light stemming reduces the over-stemming errors, but increases the under-stemming errors. On the other hand, heavy stemmers reduce the under-stemming errors while increasing the over-stemming errors.

Both Arabic root-based and stem-based algorithms suffer from generating stemming errors. The main cause of this problem is the stemmer's lack of knowledge of the word's lexical category (i.e. noun, verb, proposition, etc.)

To mitigate the drawbacks of the previous work on Arabic stemming, we propose an alternative that defines a rule to stem words instead of chopping off the letters. This rule is set by the syntactical structure of the word. For example, verbs require aggressive stemming and need to be represented by their roots. Nouns on the contrary only require light suffixes and prefixes elimination. This advanced stemming is known as Lemmatization [6].

In this work, we propose the first Arabic stemming algorithm that uses the syntactical knowledge to make stemming decisions, and we hypothesize that my approach will be more efficient in tokenizing Arabic documents than the existing approaches. In addition to the general stemming benefits, my approach can reduce the stemming errors, as well as stemming cost by reducing unnecessary stemming. An earlier version of my stemmer was introduced in and proved to improve clustering [6].

3. METHODOLOGY

Stop words (functional words or structural word list [1]) are words that either carry no meaning or are very common [23] thus do not represent the document. Stop words list usually contains prepositions, pronouns, and conjunctions.

Text processing often performs stop words removal early in the process, although there is currently no standardized list of Arabic Stop Words. The current available Arabic stop words list [9] introduces less than 200 words.

we was able to define more than 2,200 stop words and categorize them into "useful" and "useless" stop words.

Useless stop words are stop words that are used extensively and give no benefits to the subsequent words. On the contrary, useful stop words are words that can indicate the syntactical categories of the subsequent words. For example, in an English sentence such as *"I read a book yesterday,"* it is easy to realize that book is a noun and thus does not require aggressive stemming. Table 2 and 3 are examples of useful stop words.

Unfortunately, due to the early removal of the stop words, this valuable information is lost. The same scenario applies to Arabic language too. we believe that the useful stop words can help us identify nouns and verbs and direct us into the appropriate stemming. ETS automatically identifies nouns and verbs and generates global nouns and verbs dictionaries. The benefit of these dictionaries is to find similar nouns in the corpus that were used differently in other sentences. For example, in the following paragraph the word book is identified as a noun and was recognized as a noun in the following sentence.

I read a book yesterday, I love <u>books.</u>

In Table 2, a sub list of stop words preceding verbs is shown, and Table 3 presents some of the stop words preceding nouns. The stop words list was initially generated by three methods; English stop words translation, identification of common words in arbitrary Arabic documents, and manual search of synonyms to the previously identified stop words.

In the following section the ETS algorithm will be described in detail.

Table 2. Preposition Preceding Verbs

Preposition	**English**
حيثما	Wherever
كلّما	Whenever
إذا	If
عندما	When (not for question)

Table 3. Arabic circumstantial nouns indicating time and place

Preposition	**English Equivalence**
إلى	until ,near, towards ,to
أمامَ	in front of
باتجاه	On the direction of
بجانب	Aside, next to, beside
بعد	After
بينَ	Between
تحت	Below, beneath, down
حتى	Till (time and location)
خارج	Outside of
خلال	Through, during,
عبر	Through
على	Over
فوق	Above, up
في	In (time, location, duration)
قبلَ	Before
قريب	Near
منذ	since
وراءَ	Behind ,Beyond

4. ARABIC STEMMING ALGORITHM

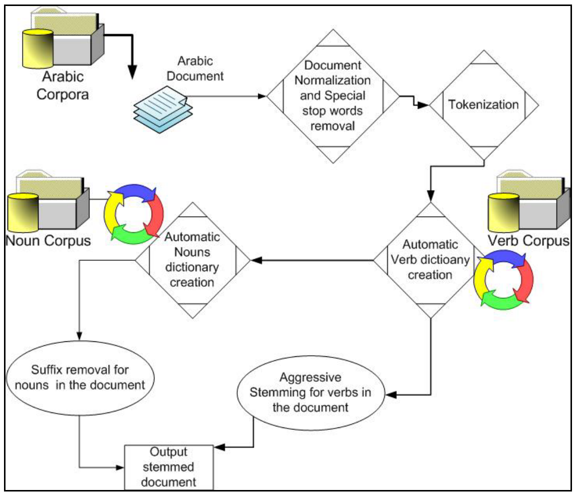
Prior to applying the ETS algorithm, normalization is performed to make the data sets more consistent. Normalization consists of the following steps:

- Convert text to Unicode.
- Remove diacritics and punctuation.
- Remove non letters (for example, numbers).
- Replace آ with double alif اا .
- Replace ى with ا .
- Replace initial إ with أ.
- Replace all hamza forms ئ , ؤ , ء with أ

As shown in Figure 4, the algorithm consists of different phases. During the first phase, useless stop words are removed to reduce the size of the corpus. Next, nouns are identified by either locating stop words that always precede nouns (example: the, a, over, etc) or words starting with definite articles. At this level, these words are flagged as nouns as a preparation for the stemming phase. In parallel to that process verbs are found by locating stop words that always precede verbs. Similar to the nouns, the verbs are added to a global verb dictionary and tagged as verbs.

In Arabic, we cannot have two consecutive verbs, thus any word following a verb is either a stop word or a noun. If the word is not a stop word then the word is added to the noun dictionary and flagged as a noun. Verbs are stemmed using the Khoja root-based stemmer and nouns are lightly stemmed. The new stemmed nouns and verbs are also added to the nouns and verb dictionary respectively.

The document is revisited by categorizing words with missing flags using the noun corpus and the verb corpus as a lookup table. Nouns usually co-occur in the same document, thus the lookup table will allow us to identify un-flagged nouns. Other words that do not belong in any category will be treated as nouns and stemmed lightly. Before we direct a word to the appropriate stemming by the word flag, all the stop words are removed since they offer no further advantage. Table 4 below summarizes the algorithm.



Arabic Corpora
Arabic Document
Document Normalization and Special stop words removal
Tokenization
Noun Corpus
Automatic Nouns dictionary creation
Automatic Verb dictioany creation
Verb Corpus
Suffix removal for nouns in the document
Aggressive Stemming for verbs in the document
Output stemmed document

Figure 4. The educated Arabic stemming algorithm simplified

Table 4. The Educated Text Stemmer (ETS) Arabic Stemming Algorithm

ETS
Input: Arabic document
Output: Stemmed document.
Noun Dictionary.
Verbs Dictionary.
V: Verb dictionary (one dimensional array sorted alphabetically[1])
N: Noun dictionary (one dimensional array sorted alphabetically)
NSW: Array of stop words proceeding nouns
VSW: Array of stop words proceeding verbs
SW: Array of stop words (including both NSW and VSW)
1. Remove useless stop words
2. Locate words attached to definite articles, and preceded by NSW and flag them as Nouns
3. Add nouns to the noun dictionary N.
4. Locate Verbs proceeded by VSW. Flag verbs in the document.
5. Add the identified Verbs to the verb dictionary V.
6. Revisit the document searching for nouns and verbs existing in the document.
7. Tokens (words) with missing tags are treated as nouns.
8. Remove the rest of the stop words (useful stop words).
9. Apply light stemming Algorithm on nouns.
10. Apply Khoja's root-based stemmer on verbs.

5. EVALUATION AND EXPERIMENTS

Different criteria are used to evaluate the performance of a stemmer. A good stemmer (by definition) is a stemmer that stems all the words to their correct roots.

Paice [3,22] introduced The stemming weight (SW) as an indicator to the stemmer efficiency. Stemming weight is the ratio between the under stemming errors and the over stemming errors.

For text mining applications that deal with a massive number of documents, the ability to significantly reduce the size of the text is also a desirable property for a good stemmer. In my previous work [6] we compared the effect of my algorithm on document clustering to that of the leading Arabic root-based stemmer presented by Khoja using Cluster Purity [24]. Applying K-means clustering on the three datasets leads to an overall cluster purity of 70.8% for the documents stemmed by ETS and 58% for the documents stemmed by Khoja's stemmer.

[1] For fast lookup, these dictionaries can be implemented using hash tables

In summary, measures discussed in the literature [21, 22, 25] to evaluate the performance of a stemmer include algorithm speed, storage saving (compression), stemming weight and retrieval effectiveness.

5.1 Paice's Evaluation Methodology

The first evaluation approach is an evaluation method proposed by Paice, applied to compare various English stemmers.

Paice introduced three new quantitative parameters to evaluate a stemmer performance: Under-stemming index (UI) , Over-stemming index(OI), and their ratio , the stemming weight (SW).

A group of morphologically and semantically related words are submitted to the stemmer. If the stemmer produces more than one stem (root for Arabic) for the same group then the stemmer has made an under-stemming error. If words belonging to different groups are stemmed to the same stem then the stemmer has made an over-stemming error. The ideal stemmer should be able to conflate (group) the related words to the same stem and has low UI and OI.

The Khoja stemmer (root-based) tends to stem morphologically related words (but not necessarily semantically related) and as a result has a high over-stemming error rate.

Arabic light stemmers only remove frequent suffixes and prefixes from the word leading to a high under-stemming error rate. To achieve a balance between these two parameters, the ratio between UI and OI, Stemming Weight (SW) is introduced.

To generate the samples to test the stemmer, we have to generate different words from a single stem by adding suffixes, infixes and prefixes. Contrary to English, in Arabic, adding affixes, prefixes or suffixes to the word can change the meaning completely. Therefore, creating a group of morphologically and semantically related words is not a trivial task. To achieve this, we first created all the possible derivations of a single stem, and then eliminated the word that does not belong semantically.

Suppose we had the following samples divided into two groups as shown in Table 5. Group 1 represents the root child and its derivations and group two represents the word parasite and it is derivations.

Table 5 : List of words derived from the stem طفل

Group 1	طفل	Child
	أطفال	Children
	الأطفال	The Children
	طفلكم	Your child
	أطفالكم	Your children
	طفولة	Childhood
	للطفولة	For childhood
Group 2	طفيلي	parasite
	طفيليات	parasites
	طفيل	parasite

The Desired merge total (DMT) is the number of different possible word form pairs in the particular group, and is given by the formula: [22,26]

$$DMT = 0.5\, n\, (n - 1)$$

Where n is the number of words in that group.

The Global Desired Merge (GDMT) is the sum of all the DMT's of the various samples.

There exists a case where certain words in a specific group can be conflated after stemming with words from another semantic group. To count all the possible word pairs formed we use the Desired Non-merge Total (DNT):

$$DNT = 0.5\, n\, (W - 1)$$

Where W is the total number of words, the sum DNT for all the groups is defined as the Global Desired Non-Merge (GDNT).

After the stemming process is performed, we would like to see if all the words in a group are conflated in the same group. To quantify the stemmer's inability to merge these words, Paice introduced the "Unachieved Merge Total" (UMT):

$$UMT = 0.5 \sum_{i=1}^{s} u_i\,(n - u_i)$$

Where s is the number of distinct stems, and ui is the number of instances of each stem.

From the sum of UMT for each group (in our example 2 groups), we obtain the Global Unachieved Merge Total (GUMT).

The under-stemming index (UI) is: GUMT/GDMT [27].

Table 6 displays the output of both stemmers. In Tables 7 and 8 the output of the ETS stemmer and the Khoja stemmer for group one and group two respectively is demonstrated.

Table 6: A comparison between the stemmers output for Sample1, the right stem is underlined.

Word	Khoja output	ETS output
طفل	طفل	طفل
أطفال	طفل	طفل
الأطفال	طفل	طفل
طفلكم	طفل	طفل
أطفالكم	طفل	طفل
طفولة	طفل	طفل
للطفولة	طفل	طفل
طفيلي	طفل	طفيل
طفيليات	طفل	طفيل
طفيل	طفل	طفيل

A stemmer might transform different words to the same stem (over-stemming). For such cases Paice introduced the Wrongly Merged Total (WMT), which is the count of over-stemming errors for each group[27].

$$WMT = 0.5\sum_{i=1}^{t} v_i(n_s - v_i)$$

Where t is the number of original groups that share the same stem, n_s is the number of instances of that stem, and, v_i is the number of stems for group t.

Similar to the above we can obtain the Global Wrongly Merged Total (GWMT) by summing the WMT for all the groups.

The over-stemming index (OI) is: GWMT/GDNT.

The table below demonstrates the evaluation of both stemmers on both groups.

Table 7: Khoja stemmer evaluation

	DMT	DNT	UMT	WMT
	24	45	0	10.5
Totals	46	139	0	10.5

Table 8: ETS stemmer evaluation

	DMT	DNT	UMT	WMT
	24	45	0	0
Totals	46	139	0	0

From the previous experiment, Khoja's stemmer had no under-stemming errors but had a UI = 0.0755. For the ETS stemmer, both UI and OI were zero.

The Khoja stemmer is an aggressive stemmer, therefore the under-stemming error was expected. We continued the experiments on many other samples, in all the cases the ETS were able to make less over-stemming errors than the Khoja stemmer. We can conclude that my stemmer is more efficient than the Khoja stemmer.

5.2 Comparison to an Expected Output

In order to assess the effectiveness of the stemmer, we compared its output to that of a manually generated stem. we randomly picked two samples of Arabic documents. The first sample contains 47 medical documents (total of 9435 words), and the second sample contains 10 long, Arabic sports articles from CNN.com (total of 7071 words).

The samples were processed manually, and words were assigned to their correct stem. In addition, a noun and a verb dictionary were created by the tester (Native Arabic speaker).

After manually generating the set of expected output, we ran my stemmer on the same set of samples. We performed these experiments at the designing phase of the ETS stemmer, thus we was able to analyze the output of the ETS stemmer and tackle the errors. we recursively repeated the experiments after adding extra rules and additional stop words.

At the end of the testing phase, on average, the ETS stemmer was able to generate 96% correct stems.

We observed that the ETS stemmer produces better results when more documents are involved in the stemming process. The automatic generation of global nouns and verbs dictionaries helps to identify nouns and verbs appearing in different context. The system initializes a new noun and verb dictionary for every experiment. The storage and the reuse of dictionaries will improve the accuracy of the stemmer where more nouns and verbs are recognized.

6. CONCLUSION AND FUTURE WORK

In this paper we introduced a novel approach for stemming Arabic documentation. we compared the performance of the new algorithm with that of the Khoja stemming algorithm using stemming weight as an evaluation criterion.

We showed that the use of presently neglected Arabic stop words can be highly effective and can provide a significant improvement when processing Arabic documents.

Additionally we introduce a new framework to normalize Arabic documents by overcoming the limitations of previous approaches, caused by the early removal of stop words.

The experiments showed a promising future for the stemming approach, which encourages further research into more in-depth comparisons of its performance with that of other leading stemmers.

7. REFERENCES

[1] G. Salton, *Automatic text processing: the transformation, analysis, and retrieval of information by computer*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[2] R.A. Baeza-Yates, "Text-Retrieval: Theory and Practice," North-Holland Publishing Co., 1992, pp. 465-476.

[3] C.D. Paice, "An evaluation method for stemming algorithms," Dublin, Ireland: Springer-Verlag New York, Inc., 1994, pp. 42-50.

[4] "Snowball: A language for stemming algorithms"; http://snowball.tartarus.org/texts/introduction.html.

[5] M. Al-Saeedi, "Awdah Almasalik ila Alfiyat Ibn Malek," *Published by Dar ihyaa al oloom {In Arabic}. Beruit, Saudi Arabia*, 1999.

[6] Eiman Al-Shammari and Jessica Lin, "A novel Arabic lemmatization algorithm," *Proceedings of the second workshop on Analytics for noisy unstructured text data*, 2008, pp. 113-118.

[7] I.A. Al-Kharashi, "Micro-AIRS: A microcomputer-based Arabic information retrieval system comparing words, stems, and roots as index terms," 1991.

[8] I.A. Al-Kharashi and M.W. Evens, "Comparing Words, Stems, and Roots as Index Terms in an Arabic Information

Retrieval System.," *Journal of the American Society for Information Science*, vol. 45, 1994, pp. 548-60.

[9] L.S. Larkey and M.E. Connell, "Arabic Information Retrieval at UMass in TREC-10," *Proceedings of the Tenth Text REtrieval Conference (TREC-10)", EM Voorhees and DK Harman ed*, 2001, pp. 562-570.

[10] L.S. Larkey, L. Ballesteros, and M.E. Connell, "Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis," Tampere, Finland: ACM, 2002, pp. 275-282.

[11] J. Xu, A. Fraser, and R. Weischedel, "Empirical studies in strategies for Arabic retrieval," *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 2002, pp. 269-274.

[12] S. Khoja and R. Garside, "Stemming Arabic Text," *Lancaster, UK, Computing Department, Lancaster University*, 1999.

[13] W. Al-Fares, "Arabic root-based clustering: An algorithm for identifying roots based on n-grams and morphological similarity," 2002.

[14] M. Aljlayl and O. Frieder, "On arabic search: improving the retrieval effectiveness via a light stemming approach," *Proceedings of the eleventh international conference on Information and knowledge management*, McLean, Virginia, USA: ACM, 2002, pp. 340-347.

[15] M. George, *Al Khaleel: A dictionary of Arabic syntax terms*, Beirut: Library of Lebanon, 1990.

[16] M.A. Al Khuli, "A Dictionary of theoretical linguistics: English-Arabic with an Arabic-English glossary," 1982.

[17] S.S. Al-Fedaghi and F. Al-Anzi, "A New Algorithm to Generate Arabic Root-Pattern Forms," *Proceedings of the 11th National Computer Conference and Exhibition*, 1989, pp. 391–400.

[18] M.A. Aljlayl, "On Arabic Search: The Effectiveness of Monolingual and Bidirectional Information Retrieval," 2002.

[19] H.K. Al Ameed et al., "Arabic Light stemmer: a new Enhanced Approach."

[20] K. Darwish, *Al-stem: A light Arabic stemmer*, 2002.

[21] C.D. Paice, "Another stemmer," *SIGIR Forum*, vol. 24, 1990, pp. 56-61.

[22] C.D. Paice, "Method for evaluation of stemming algorithms based on error counting," *Journal of the American Society for Information Science*, vol. 47, 1996, pp. 632-649.

[23] K. Lin and R. Kondadadi, "A Word-Based Soft Clustering Algorithm for Documents," *Proceedings of 16th International Conference on Computers and Their Applications*, 2001.

[24] Y. Zhao and G. Karypis, "Criterion Functions for Document Clustering," *Experiments and Analysis University of Minnesota, Department of Computer Science/Army HPC Research Center.*

[25] W.B. Frakes, "Stemming algorithms," 1992.

[26] C.D. Pake, "An Evaluation Method for Stemming Algorithms," *SIGIR'94: Proceedings of the 17th Annual International ACM-Sigir Conference on Research and Development in Information Retrieval, Dublin, Ireland, July 1994*, 1994.

[27] V.M. Orengo and C. Huyck, "A stemming algorithm for the portuguese language," *String Processing and Information Retrieval, 2001. SPIRE 2001. Proceedings. Eighth International Symposium on*, 2001, pp. 186-193.

Using English Information in Non-English Web Search

Wei Gao[*]
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong, China
wgao@se.cuhk.edu.hk

John Blitzer[*]
Computer Science, UC Berkeley
Berkeley, CA 94720-1776, USA
blitzer@cs.berkeley.edu

Ming Zhou
Microsoft Research Asia
Beijing 100190, China
mingzhou@microsoft.com

ABSTRACT
The leading web search engines have spent a decade building highly specialized ranking functions for English web pages. One of the reasons these ranking functions are effective is that they are designed around features such as PageRank, automatic query and domain taxonomies, and click-through information, etc. Unfortunately, many of these features are absent or altered in other languages. In this work, we show how to exploit these English features for a subset of Chinese queries which we call linguistically non-local (LNL). LNL Chinese queries have a minimally ambiguous English translation which also functions as a good English query. We first show how to identify pairs of Chinese LNL queries and their English counterparts from Chinese and English query logs. Then we show how to effectively exploit these pairs to improve Chinese relevance ranking. Our improved relevance ranker proceeds by (1) translating a query into English, (2) computing a cross-lingual relational graph between the Chinese and English documents, and (3) employing the relational ranking method of Qin et al. [15] to rank the Chinese documents. Our technique gives consistent improvements over a state-of-the-art Chinese mono-lingual ranker on web search data from the Microsoft Live China search engine.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval.

General Terms
Algorithms, Performance, Experimentation.

Keywords
Learning-to-rank, non-English web search, query translation, cross-lingual similarity metrics.

1. INTRODUCTION
The English web is larger and has existed longer than the web in any other language. Because of this, English search engines have been tuned for longer and with more effort than search engines in other languages. Static link analysis such as PageRank [2], click-through information [10], and query and document classification [11] are all important features for modern web search rankers. These features often don't translate directly to new languages. Because of lack of exposure, useful and important non-English sites often have low PageRank. When entering a new linguistic market, a search engine does not have a large user base, and click-through information can be unreliable. Large, reliable training sets for query and webpage classification are often unavailable in non-English languages.

At the same time, a significant portion of non-English queries have unambiguous translations into English which also function as good English queries. We designate these queries as linguistically non-local (LNL). As an example, the Chinese query "哈利波特" can be translated as "Harry Potter", a query for which abundant and useful English information exists (e.g., sites devoted to the books and movies). For this query, the features from the English documents may be able to provide us with useful information in Chinese. By contrast, the Chinese query "北方人才网" (Northern [China] skilled person network) has no immediate English counterpart. Even if we were able to accurately translate this query into English, the resulting English documents are sparse and not useful.

This work describes a method for improving search quality for linguistically non-local Chinese queries by exploiting English information. Our method falls under the framework of machine learning for search ranking [3, 4, 5, 7, 10, 15, 21]. We train our model using a list of Chinese LNL queries, together with relevance judgments for a list of Chinese documents. Training proceeds as follows: For each Chinese query, we first translate it into English and retrieve a list of English documents [6, 8]. We then use a dictionary-based translation system to compute cross-lingual similarities among the Chinese and English documents [13]. Finally, we use these cross-lingual similarities to learn a relational ranking function for the Chinese documents [15]. We show consistent improvement in relevance ranking, as measured by normalized discounted cumulative gain (NDCG) [9] on a corpus of web search data and relevance judgments from the MSN Live search engine [23, 24].

While our simple procedure does lead to improved search results, we emphasize that this is a first step, and we have by no means exhausted the possibilities for using cross-lingual information to improve search results. Perhaps most strikingly, our current method does not exploit English relevance ranking labels at all. But because English relevance ranking data is also larger and

* This work was done while the authors were visiting Microsoft Research Asia. Both authors contributed to the work equally.

Table 1. Examples of linguistically non-local (left columns) and local (right columns) Chinese queries, together with their English translations or glosses. For the linguistically non-local queries, English results may help us perform better Chinese ranking. For local queries, English results are unlikely to be helpful.

Linguistically non-local Chinese query	English translation		Local Chinese query	English gloss
福特汽车	Ford Motor Company		李白写的诗	The poems of Li-Bai
公共关系	public relations		四川长虹手机	Sichuan Changhong cell phones
哈利波特	Harry Potter		大红鹰	Great Red Eagle [Tobacco]
音乐欣赏	music appreciation		北方人才网	Northern [China] skilled person network

better constructed than other languages, this is an immediate area for further exploration. The latter part of this paper is devoted to exploring current and future approaches for using English information.

The rest of this paper is organized as follows: Section 2 introduces the concept of LNL queries; Section 3 gives a real-world LNL query example that motivates our ranking scheme; in Section 4, we present our ranking model by using relational relevance information across different languages; Section 5 discusses experiments and results; Section 6 presents the related work; and we conclude with a brief discussion of future work in Section 7 and 8.

2. LINGUISTICALLY NON-LOCAL QUERIES

Defining what makes a query linguistically non-local is a difficult problem. Because of this, we use an automatic definition derived from query logs and a large bilingual dictionary. We designate a Chinese query as linguistically non-local if its translation also occurs in the English query log. Table 1 gives several examples of linguistically local and non-local Chinese queries from the query logs of a major search engine. Even if we could translate the linguistically local Chinese queries[1], we cannot expect a large amount of rich English information (when compared to the Chinese). On the other hand, because the translations of the linguistically non-local queries occur in the English query log itself, we know a priori that they yield reasonable queries.

Even if we were able to achieve improvements on LNL queries, it would only be worthwhile if there were a significant number of them to begin with. We selected 32,730 Chinese queries and translated them into English. After automatic translation with a Chinese-English dictionary with 940,000 unique entries, we were left with 7,008 queries whose translations also appeared in an English query log of size about 7.2 million. After manually checking these queries, we found 3,767 that were perfect translations. The final ratio (3,767 / 32,730) yields the estimate that 11.5% of queries are LNL queries. We emphasize, though, that with an improved dictionary and larger query logs, this ratio may rise even higher.

[1] The linguistically local queries here did not appear in our dictionary. We provided the glosses ourselves.

3. A MOTIVATING EXAMPLE

With our taxonomy of Chinese queries as linguistically local and non-local in Section 2, we expect that the relevance ranking of Chinese LNL queries, such as foreign names, globally hot topics, general concepts, etc., will benefit from information contained in the returned documents of their corresponding English queries. For example, given the Chinese query "哈利波特" (Harry Potter), search results from the English query tend to be more relevant than just searching by Chinese because the concept is more popular in the English-speaking world.

Figure 1 shows ranking judgments for Chinese websites retrieved when given the query "哈利波特". For each retrieved document, we give a human judgment (in bold) ranging from "Bad" to "Excellent". These judgments were made independently of the English translations and this work. Similarly, if we retrieve English documents using the translation "Harry Potter", we obtain the results shown in the second column. Some documents among these results are conceptually quite similar, such as C2, E1 & E2 (all are official sites of the movie), C1 & E3 (they are about the "Sorcerer's Stone" story), and C4 & E4 (unofficial fans sites).

If we ignore inter-document similarity and use a ranker based only on the Chinese queries, we obtain the results given in the third column, where the unofficial site http://club.52harrypotter.com is ranked the highest. However, by exploiting cross-lingual similarities, we can use the fact that C1, C2, and C3 are similar to English documents which have strong PageRank, click-through, and domain recognition. From a learning-to-rank perspective, this information can be exploited to help us learn a better ranking function which increases the scores of C1, C2, and C3. Because C4 is not as similar to an official English site, its score will not be increased. Indeed, we chose this example to showcase our improved relevance ranker (shown at the rightmost column). By training a ranker which exploits document similarities, we are able to rank the websites C1, C2, and C3 above C4. The next section describes in detail how we train this ranker to improve the web search ranking for LNL Chinese queries.

Chinese Gold Standard for the query “哈利波特”	English Documents for the query “Harry Potter”	No similarity	With similarity
C1 - http://ent.sina.com.cn/m/f/f/potter1.html **(Good)** 《哈利-波特与魔法石》_影音娱乐_新浪网-片名：Harry Potter and the Sorcerer's Stone 译名：哈利・波特与魔法石/哈利・波特 1 导演：导演克里斯-哥伦布 Chris Columbus 原著：J.K.罗琳 ...	E1 - http://harrypotter.warnerbros.com/main/homepage/intro.html **(Excellent)** Harry Potter - The Official Site The Official Harry Potter Website offers content, games and activities which seamlessly extend the magical world of Harry Potter beyond the big screen…	C4	C1
C2 - http://harrypotter.tw.warnerbros.com/main/homepage/home.html **(Good)** 正式的哈利波特網站- 正式的哈利波特網站來了! 電影預告片，電影片段，霍格華茲的拍片現場，華納兄弟出品的電影哈利波特，神秘的魔法石將活生生地展現 JK 羅琳筆下的巫師和女巫，哈利波特、榮 ...	E2 - http://harrypotter.warnerbros.co.uk/diversions/index.html **(Good)** Harry Potter | Fun & Games The official site of Harry Potter! Movie trailers, film clips, behind the scenes at Hogwarts. JK Rowlings' wizards and witche’s Harry Potter, Ron Weasley, ...	C1	C3
C3 - http://www.52harrypotter.com **(Good)** 哈利波特 52Harrypotter.Com 我爱哈利波特网 新闻中心 | 同人小说 | 下载中心 | 魔法宝典 | 图库中心 | 哈利维基 | 电子期刊 | 反译联盟 | 魔法链 |丽痕书店 | 俱乐部 | 哈利热潮 哈利小说 哈利电影 哈利游戏 哈利产品 哈迷前线 哈利中国 评论反思 魔法妈妈 魔幻世界 魔幻...	E3 - http://us.imdb.com/title/tt0241527/ **(Fair)** Harry Potter and the Sorcerer's Stone (2001) - Plot summaryHarry Potter and the Sorcerer's Stone on IMDb: Movies, TV, Celebs, and more...	C3	C2
C4 - http://club.52harrypotter.com **(Bad)** 欢迎访问哈利迷俱乐部[哈利迷俱乐部] -- Powered By Dvbbs.net,20..最近没有论坛活动今日:28 帖| 昨日:1114 帖 | 最高日:3528 帖主题:6677 | 帖子:228683 | 会员:228338 | 新会员走吗喂狗 -=> 欢迎访问 哈利迷俱乐部 最新创建圈子最活跃圈子最热门圈子 52 哈利社四川分社 (创始人:冷月清霜,...	E4 - http://www.alivans.com/ **(Fair)** Magic Wands for Harry Potter wands fans - See our Magic WandsMagic Wands from Alivan's are handcrafted to meet the requirements of even the great wizard Harry Potter's wand and are similar to Harry Potter wands…	C2	C4
C5 - http://harrypotter.tw.warnerbros.com **(Bad)** Harry Potter and the Order of the Phoenix2007 Warner Bros. Ent. Harry Potter Publishing Rights © J.K.R. Harry Potter characters, names and related indicia are trademarks of and © Warner Bros. ...	E5 - http://news.bbc.co.uk/cbbcnews/hi/specials/harry_potter/ **(Bad)** CBBC Newsround | Specials | Harry PotterCBBC Newsround - Your stories, your world - first! …	C5	C5

Figure 1. Improving the search results of Chinese LNL query “哈利波特” (left-most column) by leveraging the inter-document similarity across different languages, i.e., the relationship with the information in search results of English query “Harry Potter” (second column). Enhanced ranking results can be observed in the right-most column compared to the third column.

4. A RANKING MODEL FOR LNL QUERIES

Given a set of linguistically non-local Chinese queries together with their unranked Chinese documents, we train a ranking model in three steps. First we translate each query into English and use an English search engine to obtain English documents. Then we construct a similarity graph, where nodes represent Chinese and English documents, and edges between nodes represent cross-lingual similarity. Finally we use this graph to train a relational ranking SVM [15]. This procedure is described formally in Figure 2, and the rest of this section is devoted to describing it in detail.

4.1 Query Translation

For each Chinese query, our first step is to identify a corresponding English query. Our query translation uses a large static dictionary based on a statistical query translation model [6], and we do not investigate the quality of our query translation in this work. In our experiments, we post-process our training and testing data manually to obtain Chinese-English query pairs that we are certain are correct. In deploying a real ranker, of course, we would not have the option of manually post-processing queries to ensure that they are correct valid LNL query pairs. But we emphasize that there has been significant research in the area of bilingual lexicon extraction [8], and we expect that our dictionary can be significantly improved.

4.2 Cross-lingual Similarity

Once we have obtained an English query and corresponding list of documents, the crucial next step (step 2 in Figure 2) is to determine the similarity between Chinese and English documents. We define a similarity score $sim(c,e)$ between a Chinese and English document to be a function mapping pairs of documents to a positive real number. Intuitively a good similarity measure is one which maps cross-lingual relevant documents together, and maintains a large distance between irrelevant Chinese documents and relevant English documents and vice-versa.

We use the similarity measure proposed by Mattieu et al. [13]. Using the same dictionary as for query translation, we let $T(c,e)$ indicate the set of pairs (w_c, w_e) such that w_c is a word in Chinese document c, w_e is a word in English document e, and w_e is the English translation of w_c. We define $tf(w_c, c)$ and $tf(w_e, e)$ to be the term frequency of term w_c in document c and w_e in document e, respectively. Let $df(w_c)$ be the Chinese document frequency for term w_c (with an analogous English definition). If n_c is the total number of Chinese documents, then $idf(w_c) = \log \frac{n_c}{df(w_c)}$.

```
Input:  Chinese queries and documents
$\{q_i, \{c_{ij}\}_{j=1}^{m_{ci}}\}_{i=1}^{n}$
Output: Learned ranking model which maps
from query-document pairs to real-valued
scores $f:(q,c)\to\Re$

(1)  For each query $q_i$
       Translate $q_i$ into English to
       obtain English documents $\{e_{ik}\}_{k=1}^{m_{ei}}$
(2)  For each query $q_i$
       Using a bi-lingual dictionary,
       compute a similarity adjacency
       matrix, with $\mathbf{R}_i(j,k) = sim(c_{ij}, e_{ik})$
       Let $\mathbf{L}_i = \mathbf{D}_i - \mathbf{R}_i$ be the graph
       Laplacian for $\mathbf{R}_i$
(3)  Let $\lambda$ , $\beta$ be free parameters and
     $Y_{jk}^{i} = \begin{cases} 1, & rank\ (c_{ij}) > rank\ (c_{ik}) \\ \text{-1}, & rank\ (c_{ij}) < rank\ (c_{ik}) \\ 0, & rank\ (c_{ij}) = rank\ (c_{ik}) \end{cases}$ be the entry
     of pair-wise label constraint matrix
     for query $q_i$ . Define $\hat{\mathbf{C}}_i = \mathbf{C}_i[(\mathbf{I}+\beta\mathbf{L}_i)^{-1}]^T$
     Return the solution to the
     minimization problem
     $\min_{\mathrm{f}} \|\mathrm{f}\|^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{m_{ci}} \sum_{\substack{k=j+1, \\ Y_{jk}^i \neq 0}}^{m_{ci}} \max\left[Y_{jk}^{i}\, \mathrm{f}^T(\hat{c}_{ij} - \hat{c}_{ik}) + 1, 0\right]$
```

Figure 2. Our algorithm for training a ranker for linguistically non-local Chinese queries based on RRSVM.

Bilingual idf is defined as

$$idf(w_c, w_e) = \log \frac{n_c + n_e}{df(w_c) + df(w_e)} .$$

If letting $\overline{T}(c,e)$ denote the set of terms in c that have no translation in e and likewise $\overline{T}(e,c)$ denote the set of terms in e that have no translation in c , we can define the similarity between two documents as

$$sim(c,e) = \frac{\sum_{(w_c,w_e)\in T(c,e)} tf(w_c,c)tf(w_e,e)idf(w_c,w_e)^2}{\sqrt{Z}}$$

where

$$Z = \left[\sum_{(w_c,w_e)\in T(c,e)} \big(tf(w_c,c)idf(w_c,w_e)\big)^2 + \sum_{w_c\in \overline{T}(c,e)} \big(tf(w_c,c)idf(w_c)\big)^2 \right] \times \left[\sum_{(w_c,w_e)\in T(c,e)} \big(tf(w_e,e)idf(w_c,w_e)\big)^2 + \sum_{w_e\in \overline{T}(e,c)} \big(tf(w_e,e)idf(w_e)\big)^2 \right] .$$

This similarity function can be understood as a cross-lingual analog to the commonly used mono-lingual cosine similarity function.

4.3 Relational Relevance Ranking

Once we know the most similar English documents for a particular Chinese document, we need to use these similarities to help us learn a better ranking function. The relational ranking support vector machine (RRSVM) [15] is a variant of the ranking support vector machine (RSVM) [7] that includes, in addition to the ranking objective, a constraint which encourages similar documents to have scores that are close to one another. In our case, if a Chinese document is similar to an English document with high PageRank or click-through features, the RRSVM objective will automatically encourage the Chinese document to have a higher score (provided those features are in fact useful).

Let λ and Y_{jk}^{i} be defined as in Figure 2. Here Y_{jk}^{i} is the $(j,k)^{\text{th}}$ entry of the constraint matrix for query q_i . Each entry indicates one of (-1, 0, +1) depending on whether Chinese document c_{ij} is less relevant, equally relevant, or more relevant to the query than document c_{ik} . By absorbing the margin constraints into the objective function, we can now write the ranking SVM (RSVM) objective as

$$\min_{\mathrm{f}} \|\mathrm{f}\|^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{m_{ci}} \sum_{\substack{k=j+1, \\ Y_{jk}^i \neq 0}}^{m_{ci}} \max\left[Y_{jk}^{i}\, \mathrm{f}^T(\hat{c}_{ij} - \hat{c}_{ik}) + 1, 0\right]$$

where λ is a free regularization parameter.

Now let us construct a weighted bipartite similarity graph for each query, where the edge weights for the graph are given by the similarity score $sim(c,e)$. Let the adjacency matrix be defined as in step (2) of Figure 2. For a fixed document scoring function f and query q_i , the RRSVM finds scores z_j which minimize

$$\sum_j (\mathrm{f}^T c_{ij} - z_j)^2 + \frac{\beta}{2} \sum_{j,k} \mathbf{R}_i(j,k)(z_j - z_k)^2$$

This quadratic can be solved in closed form. The solution is the minimum energy harmonic function for the graph characterized by $\mathbf{R}_i(j,k)$ [22]:

$$\mathbf{z} = (\mathbf{I} + \beta(\mathbf{D}_i - \mathbf{R}_i))^{-1} \mathbf{C}_i^T\, \mathrm{f}$$

where $\mathbf{D}_i(j,j) = \sum_k \mathbf{R}_i(j,k)$ is a diagonal matrix. The matrix $\mathbf{L}_i = \mathbf{D}_i - \mathbf{R}_i$ is the graph Laplacian for the query graph with adjacency matrix $\mathbf{R}_i$. Finally, now that we know the form of the scoring function, we may solve for the optimal f which satisfies it. This yields the optimization problem from step (3) of Figure 2.

Qin et al. [15] compute a scoring function based on ranking constraints for all of the documents corresponding to a particular query. That is, they introduce labeled constraints for every node in the graph. In contrast, we are interested only in the Chinese documents for a particular query. The English documents appear as similarity nodes in the graph, and thus in the transformation $(\mathbf{I} + \beta(\mathbf{D}_i - \mathbf{R}_i))^{-1}$, but they don't appear in the final optimization objective. Similarly at test time, we are only interested in the English documents insomuch as they influence the scores of the Chinese documents.

In Sections 2 and 3, we mentioned that our goal is to make use of English features such as PageRank and click-through that may be unavailable or unreliable in other languages. In our technique, these features appear in the transformed instance matrix $\hat{\mathbf{C}}_i = \mathbf{C}_i[(\mathbf{I} + \beta\mathbf{L}_i)^{-1}]^T$ but not in the original matrix $\mathbf{C}_i$. Unfortunately, because of the non-linear matrix inversion, this is difficult to express directly in terms of the original feature space. In general for a particular Chinese document, however, its transformed version is "smoothed" to look more like nearby English (and indirectly, Chinese) documents.

5. EXPERIMENTS AND RESULTS

In this section we give the results of a series of experiments using our proposed algorithm in web search ranking for LNL queries.

5.1 Dataset and Baselines

Because this workshop focuses on web search, all of our experiments are performed on annotated Chinese and English data of the MSN Live search engine [23, 24]. As we mentioned in Section 2, we built our training set by choosing randomly a subset of (labeled) Chinese queries from the Chinese query log and automatically translating them into English. After this, we manually choose 803 pairs of queries which are accurate translations. The average number of annotated Chinese documents for each of these queries is 25, but there is a large amount of variability (The minimum number of documents is 5 and the maximum number is 50). We end up with about 7,000 Chinese and 10,000 English documents in the data set, respectively. Each document is annotated from 0 (irrelevant) to 5 (perfect).

For each web page of a given query, the features consist of query-dependent features (e.g., term frequency) and query-independent features (e.g., PageRank) extracted from the page and the index. There are 352 such features in total.

Our baseline is a ranking SVM, trained only on Chinese documents. All of the results we report here are 4-fold cross-validated, with (approximately) 600 queries being used as training and 200 as testing. All of our results are trained using stochastic gradient descent [17] on the loss functions of the RSVM (see Section 4.3) and the RRSVM (see step (3) in Figure 2).

5.2 Mono-lingual and Joint Similarities

In Section 4, we described a bipartite graph based on the cross-lingual similarity (see Section 4.2). However, it may be that mono-lingual similarities alone can give improvement in ranking accuracy, or that a graph with both cross-lingual and mono-lingual edges can be more effective than a graph with only mono-lingual edges. We define the mono-lingual similarity between two documents to be the $(tf \times idf)$ – weighted cosine similarity. Throughout the rest of this section, mono-lingual similarity refers to a graph built only from Chinese information (and ignoring English). Cross-lingual similarity refers to bipartite graphs of the type described in Section 4, where there are Chinese-English edges, but no Chinese-Chinese edges. Joint similarity graphs are those built with both mono-lingual and cross-lingual edges.

As Section 4 indicates, there are several hyper-parameters that can be tweaked in model design. We investigate varying some of these hyper-parameters in Section 5.3. In the end, though, our goal is to compare ranking with cross-lingual and mono-lingual information. In order to do this, we select a single ranker for each of our mono-lingual, cross-lingual, and joint similarity graphs and show normalized discounted cumulative gain (NDCG) [9] for these models (see Table 2. NDCG is an IR evaluation metric that can handle multiple levels of relevance following the principles that highly relevant documents are more valuable than marginally relevant ones, and the document with a higher ranking position is more valuable because it is more likely to be examined by the user than that with a lower ranking position). Then we selected each of these results by choosing the best possible settings of hyper-parameters k (graph local neighborhood size) and β. The optimal hyper-parameters are determined by maximizing the average relative gain in NDCG. That is, for each of mono-lingual, cross-lingual and joint similarity graphs, we choose the best setting of k and β for the average relative improvement in NDCG over the no-graph baseline at different ranking positions. For a particular model m, we write $NDCG_m$ to be the model NDCG and $NDCG_{ng}$ to be the NDCG of the "no-graph" model, which ignores similarity information. Then the average relative improvement in NDCG (over ranking positions 1, 3, and 5 in our case) is defined as

$$\frac{1}{3}\frac{NDCG_m@1 - NDCG_{ng}@1}{1 - NDCG_{ng}@1} + \frac{1}{3}\frac{NDCG_m@3 - NDCG_{ng}@3}{1 - NDCG_{ng}@3} + \frac{1}{3}\frac{NDCG_m@5 - NDCG_{ng}@5}{1 - NDCG_{ng}@5}$$

Table 2. Performance of RRSVM in terms of NDCG@1,3,5 examined under different similarity measures and compared with the RSVM baseline without using the similarity graph. The optimal parameters are resolved by maximizing the average relative gains over NDCG@1,3,5

	NDCG@1	NDCG@3	NDCG@5
no-graph (baseline)	65.61	74.08	78.38
mono-lingual (k=5, β=0.4)	66.38 (+1.17%)	74.78 (+0.94%)	78.62 (+0.31%)
cross-lingual (k =20, β =0.2)	66.74 (+1.72%)	**74.81 (+0.99%)**	78.94 (+0.71%)
joint (k-all, β =0.5)	**66.9 (+1.97%)**	74.46 (+0.51%)	**78.97 (+0.75%)**

As we can see, the RRSVM consistently outperforms the no-graph baseline, which implies the effectiveness of using a similarity graph. Furthermore, at these settings using cross-lingual similarity improves over mono-lingual similarity for NDCG@1, 3, and 5. This suggests that while not perfect, the cross-lingual similarity measure does capture useful similarity information among the documents across different languages.

It is also worth noticing that the RRSVM improves NDCG@1 much better than the performance at the rest of the positions. The mono-lingual, cross-lingual and joint similarities can boost NDCG@1 above the baseline by 1.17%, 1.72% and 1.97% respectively, while the improvements at 3 and 5 are clearly less than 1%, and also NDCG@3 is basically improved larger than NDCG@5 except for using joint similarity. We don't have a good explanation for why joint similarity performs comparably worse at

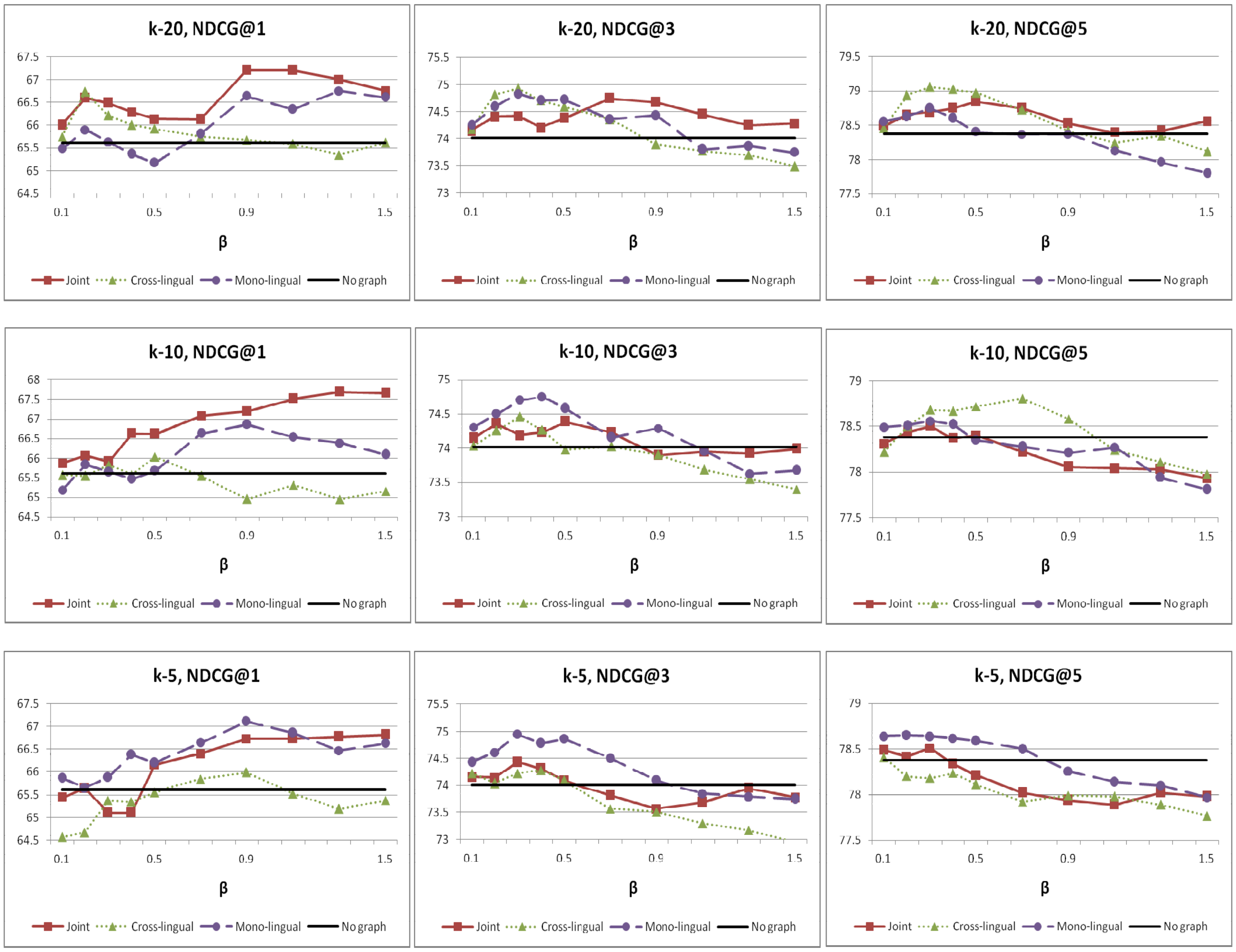


Figure 3. The comparison of different similarity measures contributing to the performance of Chinese documents ranking in terms of NDCG@1,3,5. The number of nearest neighbors *k* and the tradeoff parameter *β* are varied to show their influences.

NDCG@3, and ultimately we don't have a complete understanding of how cross-lingual and mono-lingual similarities interact. This is an important area for further investigation.

5.3 Graph Topology and β

It is often the case that in high dimensional vector spaces such as ours, local similarities can be more reliable than similarities among more distant points. Because of this, we also investigate truncating the edges for each node to its nearest neighbors. Finally, we examine how the distance changes as we increase the tradeoff parameter β (see Figure 3), which governs the relative tradeoff between the similarity graph and the ranking function. When $\beta = 0$, we use only the ranking function, and as we increase β, we increasingly penalize documents which are nearby in the graph but have dissimilar scores.

Figure 3 illustrates varying neighborhood size k (for *k*=5, 10 and 20 nodes in the graph) and β (on the x-axis). As a general trend, we observe that all of the similarities perform well under larger k. In particular, the joint similarity is consistently above the no-graph baseline when $k = 20$. Its performance degrades with fewer numbers of nearest neighbors used for the cases of NDCG@3 and 5. Cross-lingual similarity performs well under similar situations, but is less robust to smaller values of k. However, cross-lingual similarity outperforms both other methods for large k and small β, resulting in the largest overall average relative gain in NDCG. Finally, mono-lingual similarity performs best when $k = 5$.

The variance with k seems to indicate that mono-lingual similarity is accurate on a per-document basis, but overall it has limited ability to improve ranking. In contrast, our current cross-lingual similarity measure is accurate only in aggregate across many documents. Developing a way to improve cross-lingual similarity or to interpolate more accurately between cross-lingual, joint, and mono-lingual similarities is a topic for further research (see section 7). Finally, we note that while the variation of NDCG in β is fairly smooth, the graphs do not show a convex shape with a single clear maximum. Indeed, NDCG may decrease with

β before increasing again. Unfortunately, this is due to the complex nature of the inverse Laplacian and the NDCG measure itself, both of which are non-linear functions. We also plan a more thorough investigation of exactly how NDCG and other evaluation metrics interact with our hyper-parameters.

5.4 Illustrative Examples

Here we will give several illustrative examples using the queries and the ranking results from our dataset.

For example, given the query "皇家马德里" (Real Madrid), we have the Chinese website of Real Madrid Fans Club (皇家马德里球迷俱乐部|皇马中文网站, http://www.realmadridfans.com) ranked at the 6-th position by RSVM. According to our human judgments, this page should be ranked third. Using our cross-lingual similarities, we promote the site to the 4th position by RRSVM. This is because the English homepage of the "Real Madrid Fan Community" (http://www.realmadrid.dk/), labeled as "excellent", has high similarity with this Chinese page.

Another interesting example is from the Chinese query "丰田" (Toyota). Although the RSVM baseline ranks the Chinese homepage of Toyota (http://www.toyota.com.cn) as the 6-th position, RRSVM can promote it to the third. In this case, we use both mono-lingual and cross-lingual similarity information. First, the Chinese page for "Vios" (威驰, http://www.vios.com.cn), a popular automobile product of a joint-venture firm with Toyota in China, is an "excellent" result for the query and is highly similar to http://www.toyota.com.cn. Cross-lingually, Toyota's English homepage is also a nearest neighbor of the Chinese homepage.

6. RELATED WORK

Learning to rank is a broad, and in recent years very popular field. Our work does not address different mechanisms for supervised learning of ranking functions, and we cannot possibly hope to cover this entire area in detail here. We briefly mention that although classification and metric regression [4] can be used to model ranking functions, the most widely-used methods typically attempt to model a pair-wise ordering loss among documents [3, 5, 7, 10, 21]. That is, given a particular query q_i, for each pair of documents d_{ij} and d_{ik} such that d_{ij} is more relevant than d_{ik}, the loss is some function of the difference in scores between the two documents. The RSVM model [7] which we use as our baseline falls into this category. More recent methods have also investigated directly optimizing IR evaluation measures [21]. While using these methods could potentially improve our results, we decided against them for the sake of simplicity. None of these methods consider inter-document similarity as useful information. As we mentioned before, the relation ranking SVM of Qin et al. [15] considers inter-document similarity, but they consider only mono-lingual similarity among the English documents.

The other closely-related area is multi-lingual information retrieval (MLIR). The goal of MLIR systems is to simultaneously rank documents in multiple languages. In contrast, we focus on using multi-lingual information to rank documents in a *single* language. Because of this difference in goals, most existing work in MLIR focuses on heuristics for merging ranked lists from multiple languages [1, 16, 18]. While it is not our primary focus, it is possible that the techniques we outline in this paper would also be useful for multi-lingual ranking.

7. FUTURE WORK

We believe our results in Section 5 demonstrate that there is useful information available from English rankers for Chinese data. But our current algorithm has by no means exhausted the possibilities for exploiting cross-lingual information in mono-lingual search.

7.1 Improved Similarity

Perhaps the most obvious improvement is a better cross-lingual (and mono-lingual) similarity measure. Our current similarity measures do not use state-of-the-art bi-lexicon mining techniques or machine translation, and they make no attempt to distinguish text in different sections of web pages or images. At the same time, it seems natural to assume that an effective similarity score would involve some combination of page layout, varying levels of text understand and machine translation, and perhaps even image understanding. With such a large number of factors, it makes sense to consider learning a similarity function. While there has been work on learning the parameters of random walks [20], which are closely related to the Gaussian random field methods of the RRSVM [22], it is unclear if these could be directly applied. This remains an important topic for further research.

7.2 More Sources of Cross-Lingual Information

When considering the ways English information could be used in non-English ranking, we can construct a rough taxonomy of non-English queries. At a high level, there are 4 categories

(1) Linguistically local queries. The majority of Chinese queries have no corresponding useful English counterpart. But that does not mean that the information available to an English ranker is completely useless cross-lingually. It could be that some useful information could be transferred from one model to another directly.

(2) LNL queries with Chinese relevance annotated. This is the subject of this work. As we discussed in 7.1, there is still more to be done in this specific case.

(3) LNL queries with only English annotated. For most major search engines, English has many more relevance judgments than other languages. This is true for LNL queries as well. Can we develop models which exploit this information more directly in non-English search?

(4) LNL queries with neither language annotated. The vast majority of LNL queries fall into this category, where there is no direct label feedback. But we do know that since the queries correspond, the rankings should (partially) correspond as well. This is most similar to multi-view semi-supervised learning [14], and cross-lingual multi-view learning has been explored extensively in natural language processing [12, 19].

8. CONCLUSION

While English web pages continue to characterize the majority of the web, non-English search is becoming increasingly important. At the same time, the major search engines all have much better English rankers than non-English rankers. In part, we believe this to be due to the number and reliability of features like PageRank, click-through, and web page and query categorization. This work investigated exploiting English information for a subset of Chinese queries which we called linguistically non-local queries.

We showed how to use query logs and a large bilingual dictionary to improve relevance scores in Chinese for these queries. We emphasize that while our work is encouraging, we believe we have only scratched the surface of the amount of ways we can potentially exploit cross-lingual information for non-English ranking.

9. ACKNOWLEDGEMENTS

We thank Tie-Yan Liu, Cheng Niu, and Tao Qin for their helpful advice throughout the course of this work. Tao Qin also provided us with a fast implementation of RSVM, which we used in the experiments of this work.

10. REFERENCES

[1] Braschler, M. and Peters, C. Cross-Language Evaluation Forum: Objectives, Results, and Achievements. *Information Retrieval*, 7(1-2):7-31, 2004.

[2] Brin, S. and Page, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proc. WWW 1998*.

[3] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N. and Hullender, G. Learning to Rank Using Gradient Descent. In *Proc. ICML 2005*, pages 89-96.

[4] Crammer, K. and Singer, Y. PRanking with Ranking. In *Proc. NIPS 2002*.

[5] Freund, Y., Iyer, R., Schapire, R. and Singer, Y. An Efficient Boosting Algorithm for Combining Preferences. *Journal of Machine Learning Research*, 4:933-969, 2004.

[6] Gao, J. F., Nie, J.-Y., Xun, E., Zhang, J., Zhou, M., and Huang, C. Improving Query Translation for CLIR Using Statistical Models. In *Proc. ACM SIGIR 2001*, pages 96-104.

[7] Herbrich, R., Graepel, T. and Obermayer, K. Support Vector Learning for Ordinal Regression. In *Proc. ICANN 2003*, pages 97-102.

[8] Huang, F., Zhang, Y., and Vogel, S. Mining Key Phrase Translations from Web Corpora. In *Proc. EMNLP 2005*, pages 483-490.

[9] Jarvelin, K. and Kekanainen, J. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proc. ACM SIGIR 2000*, pages 41-48.

[10] Joachims, T. Optimizing Search Engines Using Clickthrough Data. In *Proc. ACM SIGKDD 2002*, pages 133-142.

[11] Kang, I. H. and Kim, G. C. Query Type Classification for Web Document Retrieval. In *Proc. ACM SIGIR 2003*, pages 64-71.

[12] Klementiev, A. and Roth, D. Weakly Supervised Named Entity Transliteration and Discovery from Multilingual Comparable Corpora. In *Proc. ACL 2006*, pages 817-824.

[13] Mattieu, B. Besancon, R., and Fluhr, C. Multilingual Document Clusters Discovery. In *Proc. Recherche d'Information Assistée par Ordinateur (RIAO) 2004*, pages 1-10.

[14] Mitchell, T. and Blum, A. Combining Labeled and Unlabeled Data with Co-training. In *Proc. Conference on Learning Theory (COLT) 1998*, pages 92-100.

[15] Qin, T., Liu, T. Y., Zhang, X. D., Wang, D. S., Xiong, W. Y., and Li, H. Learning to Rank Relational Objects and Its Application to Web Search. In *Proc. WWW 2008*, pages 407-416.

[16] Savoy, J. and Berger P. Y. Selection and Merging Strategies for Multilingual Information Retrieval. In *Proc. CLEF 2004, LNCS 0302,* 2005.

[17] Shalev-Shwartz, S., Singer, Y., and Srebro, N. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proc. ICML* 2007, pages 807-814.

[18] Si, L. and Callan, J. A. Multilingual Retrieval by Combining Multiple Multilingual Ranked Lists. In *Proc. CLEF 2005, LNCS 4022,* 2006.

[19] Snyder, B. and Barzilay, R. Unsupervised Multilingual Learning for Morphological Segmentation. In *Proc. ACL 2008*.

[20] Toutanova, K. Ng, A., and Manning, C. Learning Random Walk Models for Inducing Word Dependency Distributions. In *Proc. ICML 2004*.

[21] Yue, Y., Finley, T., Radlinski, F., and Joachims, T. A Support Vector Method for Optimizing Average Precision. In *Proc. ACM SIGIR 2007*, pages 271-278.

[22] Zhu, X., Ghahramani, Z., and Lafferty, J. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proc. ICML 2003*, pages 912-919.

[23] MSN Live Search. http://live.com.

[24] MSN Live Search Chinese. http://live.com.cn.

Mining Conceptual Graphs for Knowledge Acquisition[*]

Milagros Fernández
Dept. of Computer Science
University of Corunna
Campus de Elviña s/n
15174 A Coruña, Spain
mfgavilanes@udc.es

Eric de la Clergerie
Inria
Domaine de Voluceau,
Rocquencourt, B.P. 105
78153 Le Chesnay Cedex,
France
Eric.De_La_Clergerie@inria.fr

Manuel Vilares
Dept. of Computer Science
University of Vigo
Campus As Lagoas, s/n
32004 Ourense, Spain
vilares@uvigo.es

ABSTRACT

This work addresses the use of computational linguistic analysis techniques for conceptual graphs learning from unstructured texts. A technique including both content mining and interpretation, as well as clustering and data cleaning, is introduced. Our proposal exploits sentence structure in order to generate concept hypothese, rank them according to plausibility and select the most credible ones. It enables the knowledge acquisition task to be performed without supervision, minimizing the possibility of failing to retrieve information contained in the document, in order to extract non-taxonomic relations.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Knowledge acquisition; I.2.7 [**Natural Language Processing**]: Language parsing and understanding

General Terms

Management

Keywords

Classification,Clustering, Knowledge synthesis and visualization, Text Mining

1. INTRODUCTION

Even though research on the conceptual querying concept [10] dates from the early days of *information retrieval* (IR) research, it is surprising that, nowadays, most of practical IR systems are still based on the classic *bag of words* proposal. In effect, given that text retrieval [11] is a *natural language processing* (NLP) task, the most sensible thing would be to incorporate some of the user's knowledge and reasoning capabilities in order to improve precision in query processing.

Retrieval at the conceptual level should contribute to overcoming these limitations, leading us to more complex IR approaches. However, we should first consider an automated tool for identifying concepts from raw text, which implies having efficient techniques available for dealing with both the inherent ambiguity and flexibility of the natural language.

With things as they are, the automatic construction of practical structures from text has become an active research topic; and the reduction of both the time and effort in their development process, a tremendous need. Also, given that most of human knowledge is available in textual format, the consideration of *natural language processing* (NLP) techniques to extract the latent semantics from texts seems to be an adequate starting point to cope with the problem.

To deal with the exploitation of the linguistic structure in a text without requiring predefined knowledge of the specific domain analyzed, most authors consider a combination of robust parsing, allowing semantic relations to emerge from the text, and some kind of statistical and/or heuristic strategies in order to select the most relevant of these. On the robust parsing side, it is often argued that complex text processing is impractical on real corpus [7]. So, a popular technique is *association rule learning*, applied to retrieving term associations satisfying a minimum level of support and confidence through linguistic patterns. Formally based on a deterministic finite automaton architecture, it is the simplest and computationally most efficient strategy, which allows it to deal with very large text collections. However, it first results in excessively general parsers, which can lead to a failure to identify less commonly found grammatical structures.Moreover, its deterministic condition can mean that the system discards useful interpretations, when all the available information should be translated and considered later in a specific filtering-out task, once complementary data is available and ambiguities could effectively be solved.

Concerning the statistical/heuristic task, these methods are often applied as a complement of the parsing one with a semantic clustering purpose. The goal is to simplify the initial set of semantic links proposed by the parse, eliminating ambiguous interpretations as far as possible. Given that these techniques are based on a distributional analysis intended to be applied on large corpora, time and space

[*]This work was partially supported by the Spanish Government from research project HUM2007-66607-C04-02, and by the Autonomous Government of Galicia from research projects 05PXIC30501PN, 07SIN005206PR, the Galician Network for NLP and IR, and 'axuda para la consolidación e estruturación de grupos de investigación'.

$$P(\text{denticulées:adj})_{\text{local}(0)} = \frac{\#\text{deriv-node(denticulées:adj)}}{\#\text{deriv-cluster}} \tag{1}$$

$$P(\text{denticulées:adj})_{\text{global}(n+1)} = \frac{\Sigma_{i=1}^{n} P(\text{denticulées:adj})_{\text{local(i)}}}{\#\text{occ-denticulées}} \tag{2}$$

$$P(\text{denticulées:adj})_{\text{local}(n+1)} = \frac{P(\text{denticulées:adj})_{\text{local(n)}} P(\text{denticulées:adj})_{\text{global}(n+1)}}{\Sigma_X \quad P(\text{denticulées:X})_{\text{local(n)}} P(\text{denticulées:X})_{\text{global}(n+1)}} \tag{3}$$

Table 1: Lexical categories for the word *"denticulées"*

complexity become essential factors in their design. So, although the final system should be able to compute the frequency of n-grams of words for arbitrary values of n, most authors choose to work with bigrams using association measures known from collocation discovery, such as χ^2, pointwise mutual information or log-likelihood-ratios. In addition to this primary limitation on the value for n, these formula may also provide different or even inappropriate estimations. In effect, mutual information tends to overestimate low-frequency data while log-likelihood-ratios and χ^2 assume we are dealing with normal distributions, which is not realistic when working on texts, in which rare phenomena are common.

An alternative consists of considering a statistical model allowing an *a priori* unlimited number of states defining the probabilistic dependency. Here the classic reference is the *hidden Markov model* (HMM) that does not seem to give complete satisfaction to our requirements either. So, although HMMs can apply on arbitrary n-grams, the value of n is fixed. This firstly implies that we are assuming *stationary time*[1] and *limited history*[2] hypothesis which, for example, would not enable us to deal with long distance semantic dependencies nor recursive structures. Although we could get round this problem by considering a sufficiently high value for n, this would involve prohibitive time and space performances.

In this context, our contribution can be summarized as a proposal whose aim is to produce practical understandable results by allowing the unsupervised integration of background knowledge from complex document representations, promoting the use of *conceptual graphs* (CGs) produced automatically from text, exploiting the linguistic information available in the corpus and a sophisticated grammatical formalism, extracting the latent semantics. More in detail, we introduce a text mining strategy where primary knowledge acquisition is performed through a robust parser working on a *tree-adjoining grammar* (TAG) generated from a source *meta-grammar* (MG) [5]. Working on such a mildly context-sensitive formalism we significantly increase descriptive power in relation to *context-free grammars* (CFGs), while time and space bounds are polynomial. On the other hand, the acquisition knowledge process is not considered as deterministic, but allows us to deal with different interpretations simultaneously, integrating all viable alternatives in the final knowledge representation, in such a way that documents are represented by a structure of terms enriched by relations.

For clustering purposes, we adapt an iterative algorithm inspired by an error-mining strategy [9] developed to locate and diagnose parse and lexical errors in NLP applications. This technique enables the retrieval from a large corpora of missing, incorrect or incomplete linguistic descriptions using the frequency of n-grams of words for arbitrary values of n.

[1]probabilistic dependencies value does not change with time.

[2]probabilistic dependencies are limited to n states.

2. THE RUNNING CORPUS

We introduce our proposal from a botanic corpus describing West African flora. We concentrate on the work *"Flore du Cameroun"*, published between 1963 and 2001, which is composed of about 40 volumes in French, each volume running to about 300 pages, organized as a sequence of sections, each one dedicated to one species and following a systematic structural schema. So, sections include a descriptive part enumerating morphological aspects such as color, texture or form. This implies the presence of noun phrases, adjectives and also adverbs to express frequency and intensity, and named entities to denote dimensions.

The corpus[3] describes concepts that are related both taxonomically, for example hypernymy or `"is a"` relations, and non-taxonomically. The collection also possesses a vocabulary that is shared by most text based on this matter and is of sufficient size for our purposes.

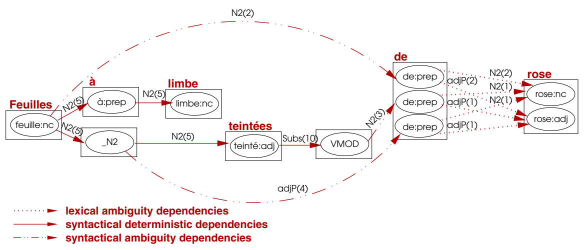


Figure 1: Parse dependencies

The present paper forms part of BIOTIM [8], a research initiative on the integral management of botanic corpus including conceptual acquisition and text mining tasks. Here, we disregard initial phases, related to the transfer from textual to electronic format [9] and also the capture of the logical structure of the text.Our linguistic starting point will be a grammar of large coverage for French and the tagged corpus. [4]

3. THE PARSING FRAME

We choose to work with TAGs [6], a mildly context-sensitive grammatical formalism that has given rise to a lot of interest in the modeling of syntax in NLP. Basically, TAGs are

[3]provided by the French Institute of Research for Cooperative Development.

[4]http://mgkit.gforge.inria.fr/

Properties	Lemmas
color	verdâtre, violacé, noirâtre, violet, jaunâtre, orange, roux, rose
form	obconique, oblancéolé, oblong, bifolié, crateriforme, punctiforme, périgone, concave, oblongoïde, ovoïde
size	moyen, petit, double, épais, inégal, entier, longue
texture	hispide, bifide, globuleux, coriace, velutineux, gélatineux, barbu
position	antérieur, dessus, voisin, seul, latéral, transversal
others	dur, bifide, frais, fréquent, jeune

Table 2: File of properties

somewhat similar to classic CFGs, but the elementary unit of rewriting is the tree rather than the symbol, allowing *extended domains of locality* (EDLs) to be specified as compared to those over which lexical constraints can be stated.

3.1 Mildly context-sensitive parsing

Any grammar formalism defines a domain of locality, that is, a domain over which various dependencies, syntactic and semantic, can be specified. This issue is related to the use of constrained systems adequate for modeling various aspects of language. In this context, the principle of EDL means that TAGs possess certain properties that make them more powerful than CFGs in terms of generative capacity. So, it allows constraints to be defined in more than one level of the parsing tree as compared to context-free rules.

Altogether, these properties lead us to conjecture that TAGs are powerful enough to model natural language while remaining efficiently parseable, but in order to fully exploit them we need an adequate operational framework. Our choice is DyALog [4], a parsing environment for a variety of grammatical formalisms, including TAGs, that returns total or partial parses. Our aim is to avoid the elimination of any parsing branch until we are sure it is not used for knowledge acquisition. In the particular case of TAGs, the system implements a parse scheme [1] verifying the VPP, which assures the best time behavior.

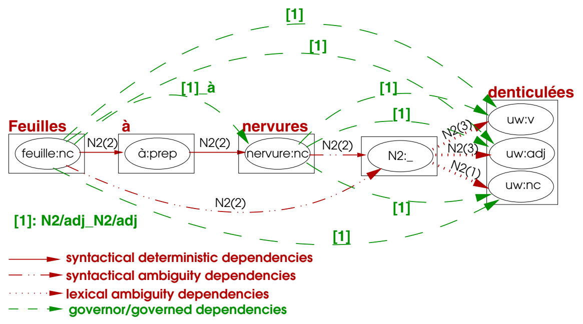


Figure 2: Another example of parse dependencies

The introduction of a high degree of abstraction in the design of the analyzer is achieved on the basis of the MG concept [5], by involving elementary constraints re-grouped in classes, these themselves inserted in a hierarchy of multiple heritage. This allows descriptions to be progressively refined, which is of particular interest when we are describing complex linguistic behavior.

3.2 Parse dependencies

The parse is resumed in a *graph of syntactic dependencies*, as is shown in Fig. 1 for the sentence *"feuilles à limbe teintées de rose"* (`"rose-tinted laminar leaves"`). Here, arrows represent binary dependencies between nodes through some syntactic construction. The parse labels each *node*, represented by an ellipse, with the tag of the corresponding lexical form, including its lemma. Rectangular shapes represent *clusters*, that is, structures referring to a position in the input string and all the possible nodes assigned by the parse at that position. So, we introduce ambiguities from both lexical and syntactic points of view. The former corresponds to clusters containing different nodes and are indicated by dotted dependencies, while syntactic ones correspond to different dependencies marked by broken dotted lines and relating to the same node.

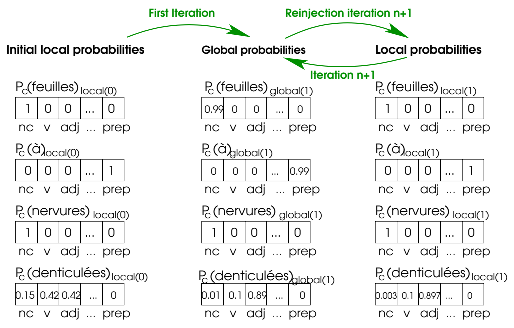


Figure 3: Computing lexical categories

This structure constitutes the starting point for detecting related knowledge and composes an initial *graph of governor/ governed dependencies*, reflecting the corresponding syntactic relationship between the nodes involved. Formally, the head of a syntagm governs its modifiers, as is shown in Fig. 2 by broken lines going from the governor node to the governed one, and labeled by functors. We shall later transform these dependencies into semantic ones.

3.2.1 *Lexical ambiguities*

Formally, the morpho-syntactic phase consists of a pipeline named Sxpipe [8] that concatenates a number of different tasks such as chunking, entity recognition and tagging.

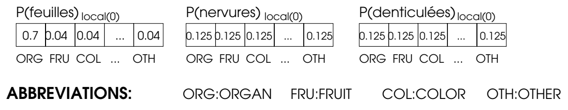


Figure 4: List of semantic weights

Tagging is often a non-deterministic and even incomplete task, especially when dealing with an encyclopedic corpus with a high degree of unknown words, that is, words whose

Word	Position	Class	Word	Position	Class	Word	Position	Class
teinte	[2]	color	couleur	[2]	color	tache	[2]	color
teinté	[2]	color	texture	[2]	texture	position	[2]	position
taille	[1,2]	size	diamètre	[1]	size	épaisseur	[1]	size
longueur	[1]	size	hauteur	[1]	size	largeur	[1]	size
altitude	[2]	size	atteindre	[2]	size	dépassant	[2]	size
atteindre	[1]	organ/fruit	forme	[2]	form	bord	[1]	organ/fruit

Table 3: File of linguistic markers

lemma is not included in the lexicon of the tagger and whose corresponding tag can only be suggested through the parser from the grammar considered. This is shown in Fig. 2, where *"denticulées"* (`"dentate"`) is labeled as an unknown word (`uw`) with three possible associated lexical categories: verb (`v`), adjective (`adj`) and common noun (`nc`). In order to avoid discarding useful interpretations, we should translate these ambiguities, which we cannot solve at a lexical level, to the syntactic phase.

This is the case of the sentence *"feuilles à limbe teintées de rose"*, that we could interpret as `"rose's tinted laminar leaves"`, as `"rose-tinted laminar leaves"` or as `"tinted laminar rose leaves"`. In the first case, *"rose"* would be a noun related to *"feuilles"* (`"leaves"`), while in the other ones it would be an adjective related to *"teintées* (`"tinted"`), as is shown in Fig. 1.

3.2.2 Syntactic ambiguities

Parsing in NLP is an incomplete task and, therefore, a source of ambiguities because it usually deals with shallow/partial strategies providing a lightweight analysis focused on identifying dependencies between nodes that are more or less close together in the text, such as noun sentences, as in *"feuilles à nervures denticulées"*, that we could locally translate in two ways: `"leaves with dentate veins"` or, alternatively, `"dentate leaves with veins"`. It here becomes impossible to establish if *"denticulées"* (`"dentate"`) relates to *"feuilles"* (`"leaves"`) or to *"nervures"* (`"veins"`), as shown in Fig. 2.

In terms of dependencies, the ambiguities, that are caused by local non-determinism, can be translated into a governor node having more than one governed node. As a consequence, to solve these ambiguities involves applying a simple syntactic constraint, namely, that a governed node should have only one governor. So, for example, in the sentence of Fig. 2, *"denticulées"* (`"dentate"`) is governed by *"feuilles"* (`"leaves"`), but also by *"nervures"* (`"veins"`) and, in consequence, we should give priority to one of these dependencies.

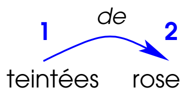


Figure 5: An example of a marked structure

No other topological restrictions are considered and, in consequence, a governor node can have more than one governed one; as in the second interpretation of Fig. 2 (`"dentate leaves with veins"`), where *"feuilles"* (`"leaves"`) is the governor for *"nervures"* (`"veins"`) and *"denticulées"* (`"dentate"`). Also, one node could be governor and governed at the same time, as in the first interpretation of Fig. 2 where *"nervures"* (`"veins"`) is the governor of *"denticulées"* (`"dentate"`), but is also governed by *"feuilles"* (`"leaves"`).

Given that ambiguous sentences require a greater use of increasing constraints to solve them, the idea behind the strategy we consider applying consists in identifying the concept that best matches the area and analyzing it, as well as its relationships with the context. This implies exploring mining techniques for discovering hidden knowledge.

4. KNOWLEDGE ACQUISITION

Once these primary syntactic dependencies have been established, probably including a number of lexical and syntactic ambiguities, our goal is to effectively extract the meaning of the corpus. This firstly implies initializing the set of classes to be considered and the dependencies between them. Later, the process will continue by compiling additional information and ranking these dependencies in order to detect those that are less plausible. Given that we are assuming our corpus is large enough, we should be able to recover this information by exploring it progressively in depth. That is, to solve the problem we only need the information we are looking for, which leads us to consider an iterative learning process in order to attain our goal.

We can illustrate this on our running corpus. So, the lexical ambiguity described in Fig. 1 should usually be decided in favor of the first alternative (`"rose's tinted laminar leaves"`), because most of us have the intuitive certainty that plants with rose colored leaves do not exist. However, this is not the case here since the rose is not a botanic species of the west Africa flora and the corpus never talks about this plant. In fact, the correct alternative is the second one (`"rose-tinted laminar leaves"`), where the word (`"tinted"`) indicates that what goes after is a color.

However, the process we are going to describe does not necessarily imply the determination of the conceptual graph. In effect, even assuming that we are working on a large corpus, we cannot be sure that a given dependency can be useless on the basis of its low weight on the graph. Perhaps the problem simply consists of the fact that some aspect of the knowledge domain is not yet fully developed in the corpus. This is a realistic hypothesis in dealing, for example, with our running botanic document collection. So, although the third alternative in Fig. 1 (`"tinted laminar rose leaves"`) is highly improbable, we should not eliminate it, but simply consider this interpretation having a low probability since we cannot discard that in the future a new variety will be unequivocally described as having rose leaves.

In this sense, our approach is inspired by an error-mining proposal originally designed to identify missing and erroneous information in parsing systems [9], combining two complementary iterative processes. For a given iteration, the first one computes, for each governor/governed pair in a

$$P(\text{feuilles:uc, [à-1], nervures:uc})_{\text{local(0)}} = \frac{\begin{array}{c} P_c(\text{feuilles:uc})_{\text{local}}\ P_c(\text{nervures:uc})_{\text{local}} \\ P_{\text{dep ini}}(\text{feuilles:uc, [à-1], nervures:uc}) \end{array}}{\Sigma_{X,Y,Z} P_c(\text{Z:X})_{\text{local}}\ P_c(\text{nervures:Y})_{\text{local}}} \quad (4)$$

$$P(\text{feuilles:uc, [à-1], nervures:uc})_{\text{global(n+1)}} = \frac{\Sigma_{i=1}^{n} P(\text{feuilles:uc, [à-1], nervures:uc})_{\text{local(i)}}}{\#\text{dep}_{\text{local(n)}}} \quad (5)$$

$$P(\text{feuilles:uc, [à-1], nervures:uc})_{\text{local(n+1)}} = \frac{\begin{array}{c} P(\text{feuilles:uc, [à-1], nervures:uc})_{\text{local(n)}} \\ P(\text{feuilles:uc, [à-1], nervures:uc})_{\text{global(n+1)}} \end{array}}{\Sigma_{X,Y,Z,T}\ \begin{array}{c} P(\text{Z:X}, T, \text{nervures:Y})_{\text{local(n)}} \\ P(\text{Z:X}, T, \text{nervures:Y})_{\text{global(n+1)}} \end{array}} \quad (6)$$

Table 4: Extraction of dependencies for *"feuilles à nervures denticulées"*

sentence, the probability of the corresponding dependency. The second process computes, from the former, the most probable semantic class to be assigned to terms involved in the dependency. So, in each iteration we look for both semantic and syntactic disambiguation, each benefiting from the other. A fixed point assures the convergence [9].

4.1 Starting the process

The first step consists in estimating, from the graph of syntactic dependencies, how a set of initial values for classes and instances can be established through the set of nodes in order to begin the iterative learning process.

Taking as our reference a list of identifiers provided by the programmer for naming the classes we are going to consider, our goal is to extract from the corpus a minimal set of nodes associated to each one. For example, in our running botanic corpus, entities could be distinguished in this list as `organ` or `fruit`, but also as properties of the type `color`, `form`, `size`, `texture` or `position`. The programmer attaches to each class a sequence of initial lemmas whose some values can be seen in Tables 2.

In future, no distinction will be considered between the terms weight, probability and preference, assuming they refer to the same statistical concept. At this point, a weight is assigned to nodes in clusters relating them with each lexical category initially associated by the parse. In order to solve ambiguities at this level, we compute them as shown in Table 1, taking as our example the word *"denticulées"*:

(1). To begin with, we compute the initial local probability for each tagged lemma of a node in a cluster, which is a simple ratio between the number of parses involving this node (`#deriv-node`) and the total number of these involving the cluster (`#deriv-cluster`). In our example, the number of parses involving the node *"denticulées"*, whose tag is an adjective (`uw:adj`), is (3), as shown with the label (`N2`) of the dependency (`N2(3)`). There are seven parses involving the corresponding cluster (`(N2(3), N2(3), N2(1)`).

(2). We re-introduce the local probabilities into the whole corpus in order to re-compute the weight of all tagged lemmas, after which we then globally estimate the most probable ones. Normalization is given by the number of occurrences of the lemma (`#occ-denticulées`), possible on different nodes.

(3). The local value in the new iteration should take into account both the global preferences and the local injection of these in the cluster, reinforcing the local probability. Normalization is given by local and global weights for the lemma, involving all possible tags (`X`) associated to the cluster considered.

We illustrate in Fig. 3 this calculus for our example in three columns, one for each step introduced. An element in these columns is a *property list of tag weights* including all tagging alternatives for the corresponding lexical form. More in detail the left-most column is the estimation of the initial local probabilities. The center one refers to the computation for the global probability, and the right-most column represents the re-injection of this in the next iteration. As we can see, in the case of *"feuilles"* the initial probability is the same as the result obtained after the first iteration because, in this sentence, *"feuilles"* has only one possible tag.

The system then associates to local occurrences of each node in the graph of governor/governed dependencies a *property list of semantic weights* reflecting the probability of the governor word being an instance of a class and the governed word to be an instance of another class, as shown in Fig. 4. Positions in one of these lists refer to probabilities for class assignment, whose sum must be equal to one, and their computation relates to the detection of particular syntactic and/or lexical patterns involving nodes with lemmas liable to be one of those instances. When this last condition is satisfied, we assign a fixed weight[5] to the corresponding entry in the property list and we equitably distribute preferences between the rest of classes in that list. For example, in the case of the word *"feuilles"*, that fixed value is put in the first position of the list of semantic weights because it is considered to be an `organ`.

With regard to the pattern recognition, we should take into account the particular characteristics of each document collection. So, in the case of our running corpus, we can assume we are dealing with a descriptive text whose kernel is composed of definitions and, therefore, dependencies related to syntactic patterns should revolve around parse structures involving nouns and/or adjectives.

On the lexical side, we take advantage of linguistic markers in order to set semantic knowledge in a context. These relations involve more explicit physical information, such as *"en forme de X"* (`"in form of X"`) or *"de couleur X"* (`"of color X"`). So, they presumably provide the most reliable information on both classes and dependencies, concentrating the vocabulary around them. We accordingly refer to the nodes and dependencies so located as *pivot nodes* and *strong dependencies*. The result serves to acquire simple

[5]in our case, this weight is 0'7.

$$P(\text{feuilles:uc:org, [à-1], nervures:uc:org})_{\text{local(0)}} = \frac{\begin{array}{l}P(\text{feuilles:uc, [à-1], nervures:uc})_{\text{local(0)}}\\ P(\text{feuilles:uc:org})_{\text{local(0)}}\\ P(\text{nervures:uc:org})_{\text{local(0)}}\end{array}}{\Sigma_{X,Y} P(\text{feuilles:uc:X})_{\text{local(0)}}\ P(\text{nervures:uc:Y})_{\text{local(0)}}} \tag{7}$$

$$\begin{array}{lr} P(\text{feuilles:uc:org, [à-1]}, X)_{\text{global(n+1)}} = \frac{\Sigma_X P(\text{feuilles:uc:org,[à-1]},X)_{\text{local(n)}}}{\#\text{dep}_{\text{local(n)}}(\text{feuilles})} & (8.1) \\ P(Y, \text{[à-1], nervures:uc:org})_{\text{global(n+1)}} = \frac{\Sigma_Y P(Y,\text{[à-1],nervures:uc:org})_{\text{local(n)}}}{\#\text{dep}_{\text{local(n)}}(\text{nervures})} & (8.2) \\ P(\text{feuilles:uc:org, [à-1], nervures:uc:org})_{\text{global(n+1)}} = \begin{array}{l} P(\text{feuilles:uc:org, [à-1]}, X)_{\text{global(n+1)}} \\ P(Y, \text{[à-1], nervures:uc:org})_{\text{global(n+1)}}\end{array} & (8.3) \end{array} \tag{8}$$

$$P(\text{feuilles:uc:org, [à-1], nervures:uc:org})_{\text{local(n+1)}} = \frac{\begin{array}{l}P(\text{feuilles:uc:org, [à-1], nervures:uc:org})_{\text{local(n)}}\\ P(\text{feuilles:uc:org, [à-1], nervures:uc:org})_{\text{global(n+1)}}\end{array}}{\Sigma_{X,Y}\ \begin{array}{l}P(\text{feuilles:uc:X, [à-1], nervures:uc:Y})_{\text{local(n)}}\\ P(\text{feuilles:uc:X, [à-1], nervures:uc:Y})_{\text{global(n+1)}}\end{array}} \tag{9}$$

Table 5: Extraction of classes for *"feuilles à nervures denticulées"*

concepts such as the value of the properties referred, or to detect enumerations that can propagate some of these values. Formally, as is shown in Table 3, we consider triples to represent markers, where the first element is the lemma playing the role of linguistic marker and the second indicates the position on the syntagm for the lemma marked by the first one. The last element is the class that will be considered as being the most probable for that marked structure.

This is illustrated that in Fig. 5 for the syntagm *"teintées de rose"* (`"rose-tinted"`), taken from the sentence *"feuilles à limbe teintées de rose"* (`"rose-tinted laminar leaves"`), where the presence of the marker *"teinté"* (`"tinted"`) indicates that the lemma *"rose"* (`"rose"`) can be embedded in the class *color*, as indicated in Table 3. As is shown in Fig. 5, the number 2 represents the position of the lemma once semantic dependencies have been extracted, as can be seen in Fig. 1, following a given syntactic pattern. Thus, *"teintées"* is considered to be in the first position, while *"rose"* is placed in the second one and both are related through a dependency labeled by the preposition *"de"*.

Once the initial process has finished, all the property lists of semantic weights associated to nodes in the graph of syntactic dependencies have been initialized, and the first assumptions on semantic dependencies between classes can be made by extending the corresponding syntactic dependencies involving nodes into these classes. The system is ready to begin with the iterative learning task, which we shall illustrate on the syntactic dependency labeled `[à_1]` relating *"feuilles"* (`"leaves"`) and *"nervures"* (`"veins"`) in Fig. 2.

4.2 Ranking of dependencies

In dealing with the ranking of dependencies, the sequence of steps to be applied by the learning process is shown in Table 4. Our aim is to associate to a probability each dependency in the graph of syntactic dependencies, denoted by `P(word1:c1,[label],word2:c2)`; with `word1` being the governor node, `c1` the lexical category of the `word1`, `label` the tag of the dependency, `word2` the governed node and `c2` the lexical category of the `word2`. More formally, we have that:

(4). We compute the local probability of the dependency in each sentence. To start the process, first tag assumptions (P_C) are provided by the error-mining algorithm [9], whose process was described in Fig. 3. We also take into account the initial probability for the dependency considered ($P_{\text{dep ini}}$), a simple ratio on all possible dependencies involving the nodes concerned. Normalization is given by the choice for the possible lexical categories, denoted by `X` and `Y`, involving each of the clusters considered as governor, expressed by `Z`.

(5). We re-introduce the local probabilities into the whole corpus in order to re-compute the weights of all possible dependencies, after which we then globally estimate the most probable ones. Normalization is given by the number of dependencies connecting the nodes considered (`#dep`).

(6). The local value in the new iteration should take into account both the global preferences and the local injection of them in the sentences, reinforcing the local probabilities. Normalization is given by previous local and global weights for the dependency, whose label is represented by `T`, involving all possible lexical categories, denoted by `X` and `Y`, associated to each of the clusters considered as governor, represented by `Z`.

4.3 Semantic class assignment

Concerning this, the sequence of steps is shown in Table 5, illustrating the computation of the probability that *"feuilles"* (`"leaves"`) and *"nervures"* (`"veins"`) are both organs, taking again the dependency labeled `[à_1]` in Fig. 2:

(7). In each sentence, we compute the local probability of this dependency if *"feuilles"* (`"leaves"`) and *"nervures"* (`"veins"`) are both organs (`org`). We start from the local weight computed in Table 4, and the initial preferences of the nodes involved in relation to class assignment as in Fig. 4. Normalization is given by the probabilities for the possible classes involving each one of the nodes considered represented by `X` and `Y`.

(8). We then calculate this preference at global level, by re-introducing it into the whole corpus in order to re-compute the weights of all the possible classes in the sentence. In order to obtain it, we first compute the probability in the whole corpus (8.1 and 8.2) for each

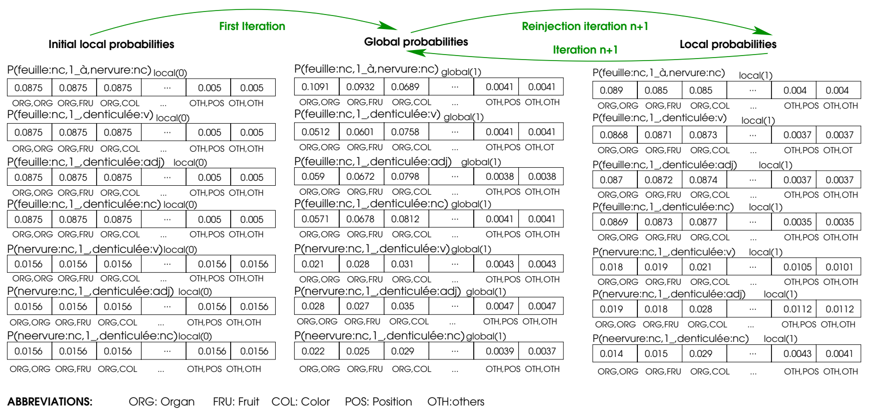

First Iteration
Reinjection iteration n+1
Iteration n+1
Initial local probabilities
Global probabilities
Local probabilities
P(feuille:nc,1_à,nervure:nc) local(0): 0.0875 0.0875 0.0875 ··· 0.005 0.005
P(feuille:nc,1_,denticulée:v) local(0): 0.0875 0.0875 0.0875 ··· 0.005 0.005
P(feuille:nc,1_,denticulée:adj) local(0): 0.0875 0.0875 0.0875 ··· 0.005 0.005
P(feuille:nc,1_,denticulée:nc) local(0): 0.0875 0.0875 0.0875 ··· 0.005 0.005
P(nervure:nc,1_,denticulée:v)local(0): 0.0156 0.0156 0.0156 ··· 0.0156 0.0156
P(nervure:nc,1_,denticulée:adj) local(0): 0.0156 0.0156 0.0156 ··· 0.0156 0.0156
P(neervure:nc,1_,denticulée:nc)local(0): 0.0156 0.0156 0.0156 ··· 0.0156 0.0156
P(feuille:nc,1_à,nervure:nc) global(1): 0.1091 0.0932 0.0689 ··· 0.0041 0.0041
P(feuille:nc,1_,denticulée:v) global(1): 0.0512 0.0601 0.0758 ··· 0.0041 0.0041
P(feuille:nc,1_,denticulée:adj) global(1): 0.059 0.0672 0.0798 ··· 0.0038 0.0038
P(feuille:nc,1_,denticulée:nc) global(1): 0.0571 0.0678 0.0812 ··· 0.0041 0.0041
P(nervure:nc,1_,denticulée:v)global(1): 0.021 0.028 0.031 ··· 0.0043 0.0043
P(nervure:nc,1_,denticulée:adj) global(1): 0.028 0.027 0.035 ··· 0.0047 0.0047
P(neervure:nc,1_,denticulée:nc)global(1): 0.022 0.025 0.029 ··· 0.0039 0.0037
P(feuille:nc,1_à,nervure:nc) local(1): 0.089 0.085 0.085 ··· 0.004 0.004
P(feuille:nc,1_,denticulée:v) local(1): 0.0868 0.0871 0.0873 ··· 0.0037 0.0037
P(feuille:nc,1_,denticulée:adj) local(1): 0.087 0.0872 0.0874 ··· 0.0037 0.0037
P(feuille:nc,1_,denticulée:nc) local(1): 0.0869 0.0873 0.0877 ··· 0.0035 0.0035
P(nervure:nc,1_,denticulée:v) local(1): 0.018 0.019 0.021 ··· 0.0105 0.0101
P(nervure:nc,1_,denticulée:adj) local(1): 0.019 0.018 0.028 ··· 0.0112 0.0112
P(neervure:nc,1_,denticulée:nc) local(1): 0.014 0.015 0.029 ··· 0.0043 0.0041
ORG,ORG ORG,FRU ORG,COL ... OTH,POS OTH,OTH
ABBREVIATIONS: ORG: Organ FRU: Fruit COL: Color POS: Position OTH:others

Figure 6: Computing semantic categories

node and semantic class, disregarding the right and left context, represented by `X` and `Y`. The final probability (8.3) is a combination of the two previous ones.

(9). After each iteration, we re-inject the previous global weight to obtain a new local one, by reinforcing the local probabilities. Normalization is done by the addition of the preferences corresponding to the nodes and classes involved in the dependency, for all the possible semantic classes considered.

We illustrate in Fig. 6 this calculus for our example in three columns, one for each step. An element in these columns is a *property list of semantic weights*. More in detail, the first column is the estimation of the initial local probabilities. So, the word *"feuilles"* (`nc`), related to the word *"nervures"* (`nc`) through a dependency labeled as `[1_à]`, has a property list where the first entry refers to the probability that *"feuilles"* and *"nervures"* could be an `organ`. The second column refers to the computation for the global probability, and the last represents the re-injection of this in the next iteration.

5. EXPERIMENTAL RESULTS

We now describe some preliminary tests on our proposal. A major drawback here, derived of the range of the corpus and the novelty and unusual of its content, is the difficulty to develop a systematic work of validation for these results, that must be done by a group of experts on west Africa flora. So, the urgency to estimate the viability of the proposal, takes us to consider a representative sampling we consider it is sufficient to provide guiding and reliable data on efficiency. We formally justify this on the uniformity of both the syntactic structure commented as well as the lexical distribution, that we show in Fig. 7. In this way, we have randomly chosen a collection of samples, each one composed by 100 sentences taken from our running corpus. In order to facilitate understanding, we focus on three of these samples, whose behavior summarizes the results obtained in all the collection.

Relating to the data compiled, when measuring the quality of an automatically created conceptual structure, the typical measures are Recall (10), Precision (11) and F-measure (12). Intuitively, *Recall* shows how much of the existing knowledge is extracted, and it is computed by

$$\text{Recall} = \frac{\#\text{correctly selected entities}}{\#\text{domain entities}} \qquad (10)$$

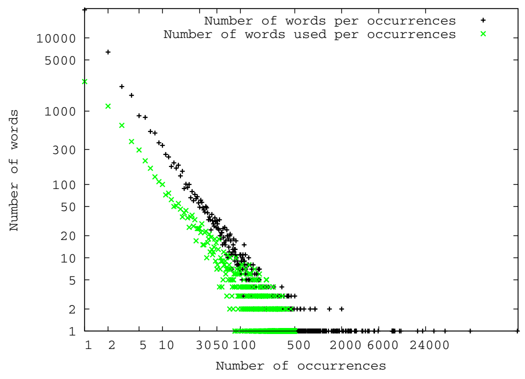


Figure 7: Word distribution in the corpus

while *Precision* specifies to which extent the knowledge is extracted correctly, and it is given by

$$\text{Precision} = \frac{\#\text{correctly selected entities}}{\#\text{total selected entities}} \qquad (11)$$

Related to the *F-measure*, it provides the weighted harmonic mean of precision and recall, summarizing the global performance of the selection process. It is computed as follows

$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \qquad (12)$$

We concentrate our attention on this last measure. Whatever is the case, these tests are performed in function of the number of iteration passes, once fixed two thresholds:

- The probability (`% success`) that a node be embedded in a class at a given moment of the process, in order to evaluate the effect of lexical non-determinism. We consider two values for testing: 20% and 90%. The former refers to a testing frame highly tolerant to this phenomena, on the contrary of the second case.
- The probability (`prob`) of a dependency when it refers to nodes shared between parses, which allows us to estimate the capability to discriminate between different interpretations, illustrating the impact of syntactic ambiguities on the learning task. We consider as values for this threshold 0'2 and 1. The former value groups dependencies on which the learning process should to

continue. The second one refers to dependencies fully identified by the algorithm, that is, considered as deterministic and, therefore, there is no reason to continue the learning process on them.

Between all the possible combinations for this set of thresholds, we focus on two extreme cases, as shown in Fig. 8. The first one illustrates a maximum level of non-determinism on both lexical and syntactic points of view. The second one compiles results for low lexical and syntactic ambiguity. Relating to the samples of sentences considered, which are divided in two groups, we baptize them as *Sample 1*, *Sample 2* and *Sample 3* in these figures for each case.

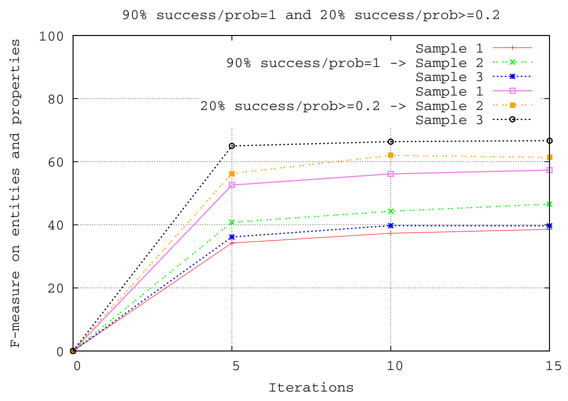


Figure 8: Maximum and minimum non-determinism

A simple analysis puts into evidence a similar qualitative behavior in the evolution of the knowledge acquisition task, with a high speed of learning that allows the process to become stabilized after only five iterations. Just as was hoped, the increasing value for the precision indexes strongly impacts the F-measure for a deterministic context as that considered in Fig. 8, for the first three samples. So, the number of detected instances of classes in this case results to be much poor in comparison with the more flexible context represented in the second three samples. In effect, intuitively, more confidence is the information relative to dependencies and node identification, and more strict are the constraints on the knowledge acquisition process. A similar reasoning can be applied in relation to the speed observing in the learning process, directly associated to the tangent of each graph, which is more steep in the second three samples.

6. CONCLUSIONS

The increasing amount of information in textual format currently available on-line is changing the way of building knowledge-based systems. On the one hand, it is inconceivable to capture it manually and, on the other, it is not possible to directly consider automatic management facilities, which has created a growing need for effective concept learning strategies. Without this kind of tools, access to relevant information runs the risk of become a frustrating and inefficient task. Our claim is that it is possible to consider the unsupervised generation of practical conceptual graphs from an unstructured corpus of sufficient size.

Our proposal attempts to reconcile quantitative and qualitative aspects on both knowledge acquisition and clustering phases. We seek to dynamically compile local, and also global, context information during the learning process. In contrast to previous works that entrust lightweight deterministic parsers with the primary knowledge acquisition task, assuming that high redundancy over a large corpus will enable mis-interpretation phenomena to be overcome; we consider an efficient non-deterministic analyzer that nips the problem in the bud. Once the corpus has been parsed and we can be sure that, sooner or later, any relevant information in the corpus will have been analyzed, do we consider filtering out useless semantic links. At this stage, we have also proved that a statistical measure based on the frequency of word sequences of arbitrary length can be used in practice to deal with semantic clustering even on very large corpora.

As a whole, preliminary experimental results seem to corroborate a promising approach as an unsupervised alternative to classic ones, but also as a possible response to solving under-specification and uncertainty problems in dealing with knowledge acquisition on unexplored domains, which could be a significant advantage for redeploying the system when no external resources are yet available.

7. REFERENCES

[1] M. Alonso, D. Cabrero, E. de la Clergerie, and M. Vilares. Tabular algorithms for TAG parsing. In *Proc. of the 9th Conf. of the European Chapter of the* ACL, pages 150–157. ACL, 1999.

[2] N. Aussenac-Gilles and J. Mothe. Ontologies as background knowledge to explore document collections. In RIAO *2004 , Avignon*, 2004.

[3] D. Bourigault, N. Aussenac-Gilles, and J. Charlet. Construction de ressources terminologiques ou ontologiques à partir de textes : un cadre unificateur pour trois études de cas. *Revue d'Intelligence Artificielle (*RIA*) M. Slodzian (Ed.)*, 18:87–110, 2004.

[4] E. de la Clergerie. DyALog: a tabular logic programming based environment for NLP. In *Proc. of 2nd Int. Workshop on Constraint Solving and Language Processing*, Barcelona, Spain, 2005.

[5] E. de la Clergerie. From metagrammars to factorized TAG/TIG parsers. In *Proc. of* IWPT'05, pages 190–191, Vancouver, Canada, Oct. 2005.

[6] A. Joshi. An introduction to Tree Adjoining Grammar. In A. Manaster-Ramer, editor, *Mathematics of Language*, pages 87–114. John Benjamins Company, 1987.

[7] Kavalec, Maedche, Svatek. Discovery of lexical entries for non-taxonomic relations in ontology learning.

[8] G. Rousse and E. de la Clergerie. Analyse automatique de documents botaniques: le project Biotim. In *Proc.* TIA'95, pages 95–104, Rouen, France, Apr. 2005.

[9] B. Sagot and E. de la Clergerie. Error mining in parsing results. In *Proc. of the 21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of the* ACL, 2006.

[10] R.C. Schank, J.L. Kolodner and G. DeJong. Conceptual information retrieval. In SIGIR *pages 94-116*, 1980.

[11] E.M. Voorhees. Natural language processing and information retrieval. In *Maria Teresa Pazienza editor,* SCIE*, pages 32-48, Lecturers Notes in Computer Science, Springer*, 1999.

Named Entity Transliteration for Cross-Language Information Retrieval using Compressed Word Format Mapping Algorithm

Srinivasan C Janarthanam
School of Informatics
University of Edinburgh
Edinburgh, UK
s.janarthanam@ed.ac.uk

Sethuramalingam S
Search and Information Extraction Lab
International Institute of Information Technology Hyderabad, AP, India
sethu@research.iiit.ac.in

Udhyakumar Nallasamy
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA
udhay@cmu.edu

ABSTRACT

Transliteration of named entities in user queries is a vital step in any Cross-Language Information Retrieval (CLIR) system. Several methods for transliteration have been proposed till date based on the nature of the languages considered. In this paper, we present a transliteration algorithm for mapping English named entities to their proper Tamil equivalents. Our algorithm employs a grapheme-based model, in which transliteration equivalents are identified by mapping the source language names to their equivalents in a target language database, instead of generating them. The basic principle is to compress the source word into its minimal form and align it across an indexed list of target language words to arrive at the top n-equivalents based on the edit distance. We compare the performance of our approach with a statistical generation approach using Microsoft Research India (MSRI) transliteration corpus. Our approach has proved very effective in terms of accuracy and time.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.7 [**Natural Language Processing**]: Text analysis

General Terms

Algorithms, Performance, Experimentation

Keywords

Cross-Language Information Retrieval, Transliteration, Compressed Word Format, Modified Levenshtein distance

1. INTRODUCTION

Transliteration refers to expressing a word in one language using the orthography of another language. Its usage is particularly common in scientific and news articles when they refer to named entities or events of another language, is different from the one used to write these articles. In this paper we present a mapping approach to transliterate named entities in English to Tamil.

The amount of Indian language content on the web has seen tremendous growth in the recent past owing to the origin of several regional news websites and information repositories like Wikipedia pages in major Indian languages Hindi, Telugu, Tamil, Bengali etc. This growth can be attributed to the increased Internet access in suburban areas and subsequent development of locally relevant content serving these communities who are more comfortable in using their native language to access such information. These multi-lingual websites can gain greater visibility in the web across the world, if they can be indexed and retrieved using normal English queries. Hence, in addition to the relevant English documents returned by a search engine, pertinent documents in languages like Tamil, Hindi, etc can also be presented to the user. In some cases, there may be more relevant documents in regional languages than in English, which is the main motivation for Cross-lingual Information Retrieval (CLIR). In order to search different languages, the CLIR system should translate the query in English into other languages and search the appropriate language archives as well.

Translating queries from English to other languages presents a lot of interesting problems. Bilingual lexicons are traditionally used to translate the queries from one language to another. However, one of the main problems with this approach is the coverage of such lexicons. As pointed by [2], out-of-lexicon words seem to constitute 83% of the highly frequent query terms. Analysis conducted by [3] revealed that around 50% of the out-of-vocabulary words were named entities. Results from [5] show that the average precision scores of a CLIR system get reduced by 50% when the named entities were not properly transliterated. Therefore accurate transliteration of named entities from a source language to a target language has significant impact in the performance of a CLIR system.

In this paper, we present an approach to transliterate named entities from English to Tamil. Apart from difference in orthography, there are several other issues with this language pair. For instance, (i) Characters in both languages do not adhere to a one-to-one mapping and (ii) Vowels and vowel digraphs in English must be aligned to appropriate short and long vowels in Tamil. In this paper we present a novel approach to transliteration in the cross language information retrieval framework. We suggest mapping instead of generating a name in the target language. We propose an algorithm to compress the given named entity in to its minimal form called Compressed Word Format (CWF) and then map it to the entries in a list of named entities in the target language. We propose two hypotheses - (i) Compressed

Word Format (CWF) mapping model is more accurate than the statistical generation models in English-to-Tamil transliteration, and (ii) In case of mapping based approaches, Compressed Word Format is more precise than the actual word forms. We have verified these hypotheses in this paper. From the evaluation results, we found that by compressing the words to their minimal forms, we were able to make the mapping process faster and more accurate. We also found that the mapping method was more accurate than the statistical generation approach.

In section 2, we briefly discuss the related work in named entity transliteration. In section 3, we present the algorithms related to our approach. We discuss our development process in section 4 and our evaluation and results in section 5.

2. PREVIOUS WORK

Transliteration of words from a source language (SL) to a target language (TL) is mostly considered to be a problem of generation of target language equivalents using statistical approaches. The major techniques for transliteration can be broadly classified into two categories namely Grapheme-based and Phoneme-based. Techniques like n-grams [5] and Finite State Machines [1] [8] were explored for transliteration of English names to Arabic and Japanese. Chinese orthography for transliterating English words into Chinese was used in [12]. In the Indian language context, a word origin based approach for splitting Indian and foreign origin words and transliterating them based on their phoneme equivalents was shown by [11]. A discriminative, CRF-HMM model for transliterating words from Hindi to English was used in [4]. Based on their transliteration system efficiency, they have shown improvements in the performance of their CLIR system for Hindi-English queries from Cross-Language Evaluation Forum (CLEF) [1].

The concept of splitting words into their composite consonants and vowels for transliteration was used earlier by [6]. They used a grapheme-based model for English to Persian Transliteration. Character alignments were based on probabilities obtained from GIZA++ [10]. Their methodology involves splitting of words into their constituent consonants & vowels from the source and target languages and focus on combining various combinations of vowels and consonants and coming up with the most probable ones.

Our approach of transliteration based on mapping can be compared to the approach followed by [7] for Arabic to English transliteration of proper Nouns. Their main focus is on building a transliteration model for Arabic-English machine translation system. They have used a Noisy-channel model for deriving English equivalents based on the alignment probabilities of English and Arabic characters. A consonant-based mapping scheme is used to compare the generated English words with their large collection of English names. The final score for the best English transliteration is generated based on the sum of scores at the three different phases of their system. Our scenario and approach differs from theirs in two ways, (1) their transliteration system is basically for machine translation where only one possible transliteration is valid. Our system is built for the CLIR scenario where one fixed transliteration may not be sufficient, (2) we have used a simple, heuristic based mapping of English and Tamil characters when compared to their HMM based Arabic-English characters alignment model.

3. OUR APPROACH

Named entities in English are mapped to their proper Tamil equivalents by comparing their minimal consonant skeletal forms or Compressed Word Format (CWF), produced based on a set of linguistic rules. Prior to it, pre-processing on the list of Tamil names has to be performed. Once we derive the compressed word formats for the English query word and for the list of compressed Tamil words equivalents, we search and match the right equivalent in the index based on our modified Levenshtein algorithm.

3.1 Pre-processing

Before the actual transliteration process, our approach involves certain pre-processing operations. Firstly, the named entities in the target language, Tamil, are collected and listed. This can ideally be done by running named entity recognizers on the archives. These names are then romanized so that they can be easily compared to the English queries. There are several romanization schemes. In most of these schemes, each Tamil character (a vowel or a consonant) is mapped to one or more Roman characters, in order to increase readability. For instance, 'aa', 'ii', etc are used for long vowels. We used a custom romanization scheme (given in the Appendix) that maps every Tamil character to Roman characters in a one-to-one mapping fashion. This mapping strategy ensures that the Romanized forms are compact and avoids any state-based processing. Secondly, using the CWF algorithm (given below), we derive the compressed word format of every word in the list. Finally, an index of tuples is created, with each tuple containing the compressed string and the actual string.

3.2 CWF algorithm

In the CWF algorithm, we compress both English and Tamil named entities from their actual forms (AF). Compressed Word Format of words is created using an ordered set of rewrite and remove rules. Rewrite rules replace characters and clusters of characters with other characters or clusters. Remove rules simply remove the characters or clusters. This algorithm is used for both English and Tamil names, but their rule sets are different. A set of English rules are given with examples.

Step 1: Rewrite consonant digraphs (group of consonants with no intervening vowels) representing one phoneme as single character consonants. This replacement is done to reduce edit distance between CWF forms.

e.g. chidhambaram → cidambaram

Step 2: Rewrite double consonants like 'kk', 'rr', etc as respective single consonant.

e.g. musharraf → musaraf

Step 3: Rewrite clusters based on target language heuristics. In our current work, we considered specific heuristics for Tamil.

Step 4: Remove the vowels. Our final CWF forms will only have the minimal consonant skeleton.

e.g. cidambram → cdmbrm

[1] CLEF - http://www.clef-campaign.org

Similar rules are developed for reducing English names to CWF format.

3.3 Modified Levenshtein algorithm

Levenshtein's Edit Distance algorithm [9] is popularly used to calculate the edit distance between any two strings in the same language. In our current work, we use it to calculate the distance between the source language string and the target language string in CWF format. We identified that using a strict match when calculating the edit distance resulted in a sub-optimal performance. Romanized Tamil strings only use a subset of Roman alphabet. For instance, characters like 'b', 'C' are not used. Furthermore, the uppercase and lowercase characters are treated differently. Also, some roman characters (like 'z') used in Tamil strings do not sound acoustically like 'z' in English strings. To accommodate these discrepancies, we relaxed the strict matching in the Levenshtein algorithm to handle phonetically equivalent characters in both languages. For instance, 'd' in English and 'T' in Tamil are considered equivalent. These pairs of characters were identified during the development cycle. We refer to this algorithm as Modified Levenshtein (MLev) algorithm in this paper.

3.4 CWF mapping algorithm

CWF Mapping algorithm is our transliteration algorithm that takes as input a source language named entity string (e.g. Chidambaram) and produces a ranked list of transliterated names in the target language (Tamil). The steps involved are given below.

Step 1: Derive the compressed word form (CWF_E) of the input string. This is obtained using the CWF algorithm (for English).

e.g. Chidambaram $\rightarrow$ cdmbrm

Here, we see how the "Ch" is replaced with "c" and that all the vowels have been removed.

Step 2: Compute the edit distance of the CWF_E with all the CWF_T Tamil entries in the index. Collect all the matches and their edit distances.

e.g. (cTmprm, ciTamparam):0, etc..

Edit distance is computed using a modified Levenshtein algorithm (MLev) We call the edit distance calculated using CWF forms and MLev algorithm as CWF+MLev.

Step 3: Rank the candidate matches based on the CWF+MLev edit distance. When there are more than one candidate at the same edit distance, finer ranking can be made using edit distance between the actual forms of source and target strings using our Modified Levenshtein algorithm (AF+MLev).

4. DEVELOPMENT PHASE

Development phase refers to the two one-time processes of designing the algorithms. One is to collect the heuristic remove and rewrite rules (RR rules) for the CWF algorithm and another is to identify the acoustically equivalent (AE) pairs for the modified Levenshtein (MLev) algorithm.

An initial set of RR rules and AE pairs were intuitively hand-coded. We then added to this initial set based on our development data during the development phase. For this purpose we created the development datasets. One of the authors manually transliterated around 7200 Indian names from Tamil to English. These names were a part of state-wise, electoral data containing the list of voters' names, provided online by the Election Commission of India[2]. We also collected around 700 foreign names with their English equivalents from Tamil Wikipedia[3] pages. At the end, we had around 8000 matching pairs in our development datasets. Each matching pair had the English named entity string and its matching Tamil string.

We ran our CWF and MLev algorithm on the two development datasets to calculate the average edit distance between matching pairs. Ideally, our edit distance algorithm must calculate the distance between matching pairs to be zero (because they perfectly match according to human annotation standards). However with the initial set of heuristics, the results were not even close, because the strings were using different character sets. We manually inspected those pairs with high edit distances between them and identified RR rules and AE pairs that need to be added to avoid the extra distance between them. For instance, replacing 'ch' with 'c' (in English strings) reduces the distance as it is transliterated to 'c' in Tamil strings. Similarly, adding 'd' and 'T' as equivalent pairs helps in reducing the distance, as they are acoustically similar although orthographically different.

We have compared our CWF+MLev scores with the standard Levenshtein edit distance scores between the actual forms of the strings (AF+Lev). The results of the experiment are shown in Table 1.

Table 1. Average Edit distance CWF+MLev vs. AF+LEV

Dataset	CWF+MLev distance	AF+LEV distance
Indian names	0.1845	4.6186
Foreign names	0.7922	6.9532

At the end of the development phase, we had an algorithm that gives the lowest edit distance for the matching pairs. The above table shows the average CWF+MLev edit distance between matching pairs, which is closer to zero than the standard Levenshtein edit distance (AF+Lev) between actual forms. The implication of this result is that, at distance 1 or less, our algorithm is more likely to search the index and produce a matching pair by using CWF+MLev distance than AF+Lev distance. For AF+Lev distance to produce a matching pair, the distance threshold has to be set to 7 (as its average distance is 6.9 for foreign names). However, at the distance threshold 7, the algorithm will pick up more spurious false matches along with the real matching string. Hence the precision will degrade although we achieve a high recall. However, with our approach of combining MLev algorithm with CWF word forms, we can produce high recall at shorter distance thresholds and therefore retain high precision.

[2] ECI list - http://www.eci.gov.in/DevForum/fullname.asp

[3] Tamil Wikipedia - http://ta.wikipedia.org

5. EVALUATION

In order to prove our hypotheses, we ran two experiments. In the first experiment, we compared the performance of CWF Mapping algorithm against IIIT's CRF+HMM Generation Model [4]. In the second experiment, we compared the performance of our algorithm against the standard Levenshtein algorithm. For both these experiments, we used a parallel dataset containing names in five Indian languages provided by Microsoft Research India. It consists of 3300 names (3000 in the training set and 300 in the testing set) parallel in English, Tamil and other Indian languages. We paired the English names with their corresponding Tamil names in both training and testing lists. We created an index of Tamil names (as described in Section 3.1) using both the training and testing lists.

5.1 First experiment

We ran our CWF Mapping algorithm on the 300 names from the test set. The algorithm found potential matches for all the 300 names from the index of 3300 names and output a ranked list for each input English name. The ranking is done based on the edit distance between the query and the candidate, in such a way that the candidate with the lowest edit distance is ranked high, and so on. However in the ranked list, sometimes, there is more than one candidate per rank because they all have the same edit distance. This makes it difficult for us to compare our results to other algorithms that produce a list with one candidate per rank and usually report their accuracy at top 1, 2, 5 and 10 candidates.

Table 2 shows the potential candidates for the English input query 'Bakul' at top 1, 2, 5 and 10 ranks. At top 1 rank (which is just rank 1), there are 2 potential candidates. At top 2 ranks (which consists of candidates from Rank 1 and 2, we have 4 candidates) and so on. This illustrates our case of not having one candidate per rank.

Table 2. Ranked list of candidates for the English input query name ' bakul', which has 'pakUl' as the correct romanized Tamil equivalent

Top N Rank	No. of candidates	Candidates
Top 1 rank	2	pAkul; pakUl;
Top 2 ranks	4	pAkul; pakUl; pOkil; pOklE;
Top 5 ranks	11	pAkul; pakUl; pOkil; pOklE; kul; kulE; pAkE; pAlE; pAkki; patkal; uppal;
Top 10 ranks	24	pAkul; pakUl; pOkil; pOklE; kul; kulE; pAkE; pAlE; pAkki; patkal; uppal; kAl; AkalE; akOlA; kOlE; pIlA; kellA; pOkkE; pellO; pikki; pinkal; qpilA; piGkAlA; pinkalE;

Hence, in order to provide a fair comparison, we tried to figure out the average number of candidates that are identified for top n ranks over the test set. This is shown in Table 3.

Table 3. Average number of candidates

Top n ranks	1	2	5	10
Ave. No. of candidates	1.10	2.38	7.87	20.13

From the above table, we see that there are on average 1, 3, 8 and 20 (rounded) candidates at top 1, 2, 5, and 10 ranks. Therefore, we report our accuracy results at top 1, 2, 5 and 10 ranks and compare them with top 1, 3, 8 and 20 candidates from the generation algorithm. Table 4, reports the accuracy of our algorithm at top n ranks.

Table 4. Ranked list of candidates for the entire 300 English query names in the MSRI test data

Top n rank	Number of queries	Accuracy
1	271	0.91
2	280	0.94
5	287	0.96
10	295	0.99

Out of 300 query names, the exact match was found to be in Top 1 rank (the first candidate) for 271 query names. Similarly, the exact match was found for 280 query names at top 2 ranks (in the first 3 candidates), for 287 query names at top 5 ranks (in the first 8 candidates), and for 295 query names at top 10 ranks (in the first 20 candidates).

In the following table (Table 5) we compare our results with the CRF-HMM generation model accuracy results at top 1, 3, 8 and 20 candidates.

Table 5. Transliteration accuracy of our CWF mapping algorithm compared with IIIT's CRF-HMM model

System	Top 1	Top 3	Top 8	Top 20
CRF-HMM	0.42	0.62	0.77	0.83
CWF Mapping	0.91	0.94	0.96	0.99

The accuracy scores in Table 5 clearly show that our mapping based technique clearly outperforms the IIIT's statistical generative model for English-Tamil transliteration by a large margin. Figure 1 shows the improvement in transliteration accuracy between the two systems at different ranks. This proves our first hypothesis that mapping based approach is more efficient than the statistical transliteration approach for English-Tamil transliteration.

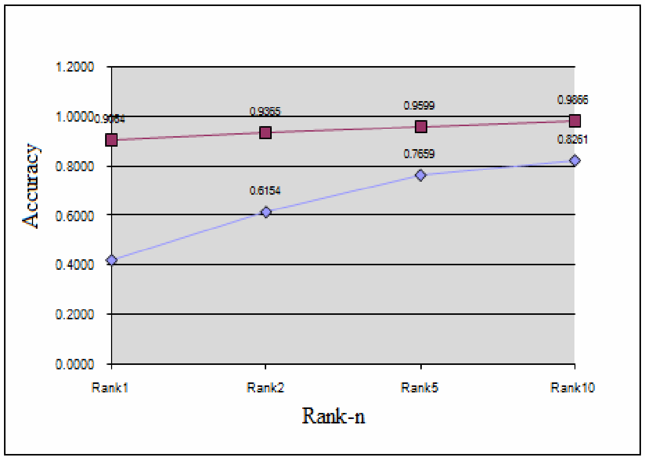


Figure 1. Transliteration performance comparison

On analysis, we found that most of the candidates generated by the IIIT's generation algorithm were not even found in our index. This has significantly reduced the accuracy of the algorithm. In order to improve the accuracy, we mapped each of the generated candidates to its closest (in terms of standard Levenshtein distance) match in the index. We did not use modified Levenshtein distance here, because the generated candidates that are mapped to the index entries are already in Tamil Romanized form and MLev algorithm needs one string from English and another from Tamil. This mapping step ensures that each candidate in the list is now in the index as well. This additional step increased the accuracy of the IIIT algorithm, proving again that mapping approach is more beneficial. We then compared the results of CRF-HMM model (now enhanced by mapping) to our results. This is given in Table 6.

Table 6. Accuracy of CWF mapping algorithm compared with modified IIIT's CRF-HMM model

Top n candidates / Model	**1 (Top 1)**	**3 (Top 2)**	**8 (Top 5)**	**20 (Top 10)**
Modified CRF-HMM	0.79	0.91	0.94	0.94
CWF Mapping	*0.91*	*0.94*	*0.96*	*0.99*

The above results show that our mapping algorithm still performs significantly better than Generate and Map model. In addition to accuracy, the CWF Mapping model provides an advantage of lesser execution time than the Generate and Map model. In order to map 20 generated candidates to names in the index, we had to run Levenshtein's algorithm 20 * 3300 times (named entity index size is 3300 names). And this was run for every query in the 300 name test dataset. In contrast, in our mapping model, we only have to run our modified Levenshtein (MLev) algorithm 3300 times to identify the top 20 candidates. In addition, for each of these 20 candidates, we had to run Levenshtein algorithm between the actual forms of the pairs for the purpose of finer ranking, so 3300 + 20 iterations in total. Hence for m entries in the index and n candidates, the IIIT's generate and map model executes Levenshtein algorithm (n * m) times, while our mapping model only executes it n + m times. Also, since after compression the strings are only half as long, the modified Levenshtein algorithm takes only half the time as the standard Levenshtein algorithm. Reduction in execution time becomes a significant advantage as index gets larger.

5.2 Second experiment

In our second experiment, we compared two versions of our mapping algorithm. One used the CWF+MLev distance and another used AF+Lev distance for ranking the candidates. This experiment was done to demonstrate the effect of CWF and MLev algorithms in our mapping model. We evaluated them on the MSRI test dataset (300 names) and the index containing 3300 names. The following table (Table 6) shows how accurate the algorithm is by using the two edit distances. In the parentheses, we mention the average number of candidates.

Table 7. Transliteration accuracy of our compressed word format, modified Levenshtein algorithm (CWF+MLev) compared with normal Levenshtein algorithm

Top n candidates	**CWF+MLev**	**Levenshtein**
Top 1 rank	0.91 (1)	0.60 (12)
Top 2 ranks	0.94 (3)	0.88 (141)
Top 5 ranks	0.96 (8)	1 (3002)
Top 10 ranks	0.99 (20)	1 (3004)

The comparative results in the Table 6 shows that Compressed Word Format distance based mapping helps in achieving far better results when compared to the normal Levenshtein edit distance. One should note that there are 12 competing candidates at rank 1, while using AF+Lev distance instead of CWF+MLev distance. This means that, although 59% of queries found their exact match in rank 1, there are 11 other competing candidate and we have no way to tell which of the 12 candidates is the exact match. Precision is hugely sacrificed to improve recall. Similarly, we have 141, 3002 and 3004 candidates at top 2, 5 and 10 ranks respectively. Although there is 100% recall at top 5 ranks, the number of candidates mapped is 3002. We cannot possibly use 3002 candidates to search the Tamil archives. Considering the loss of precision and high accuracy at top 1 and 2 ranks, we could conclude that the mapping algorithm using CWF+MLev distance is more effective than the one using AF+Lev distance. This validates our second hypothesis that compression improves accuracy and precision in the mapping models.

6. CONCLUSION

In this paper we have presented an efficient algorithm for transliteration of English named entities to Tamil. We identified the challenges in transliteration in the language pair. We then proposed the CWF algorithm where strings were compressed without losing essential information needed for transliteration. We have shown how to use compressed word forms and modified Levenshtein algorithms to produce accurate transliterations of named entities in the target language. Our results have proved that mapping strategy is a better option than generating words in the context of transliteration. Our results show that by combining Compressed Word Format (CWF) with modified Levenshtein algorithm, we can bring the best of both worlds together, i.e. increase accuracy without sacrificing precision.

7. FUTURE WORK

Although CWF forms reduce the execution time of modified Levenshtein algorithms, with increasing size of the index, the execution time could get significantly longer. We plan to work on ways to make the search faster using Finite State Transducers (FSTs). We would also like to automatically learn the remove and rewrite rules from an aligned corpus using statistical machine learning techniques. We plan to integrate our transliteration model to an English-Tamil CLIR system and empirically evaluate the effect of compression and mapping on document retrieval.

8. ACKNOWLEDGMENTS

Our sincere thanks to Mr. Prasad Pingali and Mr. Surya Ganesh of IIIT-Hyderabad for sharing their results with us and to Dr.A.Kumaran, Microsoft Research India for providing the Indian language transliteration parallel corpus for our experiments.

9. REFERENCES

[1] Al-Onaizan, Y. and Knight, K. 2002. Machine transliteration of names in Arabic text. In Proceedings of the ACL-02 Workshop on Computational Approaches To Semitic Languages (Philadelphia, Pennsylvania). Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 1-13.

[2] Cheng, P.J., Teng, J.W., Chen, R.C., Wang J.H., Lu, W.H. and Chien, L.F. 2004. Translating unknown queries with web corpora for cross-language information retrieval. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, July 25-29, 2004, Sheffield, UK.

[3] Davis, M. W., and Ogden, W. C. 1997. Free resources and advanced alignment for cross-language text retrieval. In Proceedings of The Sixth Text Retrieval Conference (TREC-6). NIST, Gaithersbury, MD, USA.

[4] Ganesh, S., Harsha, S., Pingali, P. and Varma, V. 2008. Statistical Transliteration for Cross Language Information Retrieval using HMM aligment and CRF, In Proceedings of International Joint Conference on Natural Language Processing(?CNLP)-2008, NERSSEAL Workshop, Hyderabad, India.

[5] Jaleel, N. A. and Larkey, L.S. Statistical transliteration for English-Arabic cross language information retrieval. In Proceedings of the twelfth international conference on Information and knowledge management, November, 2003, New Orleans, LA, USA.

[6] Karimi, S., Turpin, A. and Scholer, F. 2006. English to Persian Transliteration. In Proceedings of Symposium on String Processing and Information Retrieval (SPIRE), October 2006, Glasgow, UK, pp. 255-266.

[7] Kashani, M.M., Popowich, F., and Sadat, F. Automatic transliteration of proper nouns from Arabic to English. The Challenge of Arabic for NLP/MT. In Proceddings of International conference at the British Computer Society, October 2006, London, UK, pp.76-83.

[8] Knight, K. and Graehl, Jonathan. 1997. Machine transliteration. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, pp. 128-135. Morgan Kaufmann.

[9] Levenshtein, V.I. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10 (1966):707–710.

[10] Och, F.J. and Ney, H. 2003. A systematic comparison of various statistical alignment models, Computational Linguistics, volume 29, number 1, pp. 19-51 March 2003

[11] Surana, H. and Singh, A.K. 2008. A More Discerning and Adaptable Multilingual Transliteration Mechanism for Indian Languages. In Proceedings of International Joint Conference on Natural Language Processing(IJCNLP)-2008, Hyderabad, India.

[12] Virga, P. and Khudanpur, S. 2003. Transliteration of proper names in cross-lingual information retrieval. In Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-Language Named Entity Recognition - Volume 15 Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 57-64. DOI= http://dx.doi.org/10.3115/1119384.1119392.

10. APPENDIX

Our Romanization scheme showing one-to-one mapping between Tamil characters and the Roman characters is given below.

அ	a	க	k	வ	v
ஆ	A	ங	G	ழ	z
இ	i	ச	c	ள	L
ஈ	I	ஞ	J	ற	R
உ	u	ட	t	ன	n
ஊ	U	ண	N	ஜ	j
எ	e	த	T	ஷ	Z
ஏ	E	ந	D	ஸ	S
ஐ	Y	ப	p	ஹ	h
ஒ	o	ம	m		
ஓ	O	ய	y		
ஔ	W	ர	r		
ஃ	q	ல	l		

Corrupted Queries in Spanish Text Retrieval: Error Correction vs. N-Grams[*]

Juan Otero
Dept. of Computer Science
University of Vigo
Campus As Lagoas, s/n
32004 Ourense, Spain
jop@uvigo.es

Jesús Vilares
Dept. of Computer Science
University of A Coruña
Campus de Elviña s/n
15174 A Coruña, Spain
jvilares@udc.es

Manuel Vilares
Dept. of Computer Science
University of Vigo
Campus As Lagoas, s/n
32004 Ourense, Spain
vilares@uvigo.es

ABSTRACT

In this paper, we propose and evaluate two different alternatives to deal with degraded queries on Spanish IR applications. The first one is an n-gram-based strategy which has no dependence on the degree of available linguistic knowledge. On the other hand, we propose two spelling correction techniques, one of which has a strong dependence on a stochastic model that must be previously built from a POS-tagged corpus. In order to study their validity, a testing framework has been formally designed and applied on both approaches.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Linguistic processing*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Natural language*

General Terms

Experimentation, performance

Keywords

Character n-grams, degraded queries, information retrieval, spelling correction

[*]This work has been partially supported by the Spanish Government from research projects HUM2007-66607-C04-02 and HUM2007-66607-C04-03, and by the Autonomous Government of Galicia from research projects 05PXIC30501PN, 07SIN005206PR, the Galician Network for NLP and IR and *"Axuda para a consolidación e estruturación de unidades de investigación"*.

1. INTRODUCTION

Although formal models for *information retrieval* (IR) are designed for well-spelled corpora and queries, useful questioning should be robust against corruption phenomena and out-of-vocabulary words, in order to avoid increasing silence due to missing information, imprecision, inconsistency or uncertainty. So, tolerant retrieval becomes a priority in the design of IR applications, particularly when we are dealing with highly dynamic databases that continuously change over time, perhaps making machine-discovered knowledge inconsistent. More intuitively, we could say that such imperfection is a fact of life in this kind of systems, now largely popularized through web services.

A major factor at the root of these problems is the introduction of spelling errors by the user, either by accident, or because the term he is searching for has no unambiguous spelling in the collection. It is therefore imperative to study this problem in query languages, since it can substantially hinder performance. In this sense, most authors directly apply error correction techniques on lexical forms in order to provide IR tools with a robust querying facility.

This strategy, often considered in the domain of *natural language processing* (NLP) in order to analyze degraded texts, possesses in this case an unusual feature. In effect, while common NLP tools tolerate lower first guess accuracy in dealing with error correction by returning multiple guesses and allowing the user to interact with the system in order to make the final choice of the correction, this is not common in IR systems, thus increasing the complexity of the problem. In fact, although enhanced string matching algorithms for corrupted text have been introduced in order to improve recall while keeping precision at acceptable levels [4], the absence of interactivity with the user has a negative impact on their effectiveness.

In practice, spelling correction proposals [21] apply modifications on the strings in order to minimize the *edit distance* [9] between them; that is, the number of edit operations[1] to be considered in order to transform one of these strings into the other. Usually this concept is extended by assigning different weights to different kinds of edit operations, responding to some kind of linguistic criteria. In this way, a first attempt [22] consists of introducing term weighting functions to assign importance to the individual words of a document representation, in such a manner that it can

[1]Insertion, deletion or replacement of a character, or transposition of two contiguous characters.

be more or less dependent on the level of corruption. A complementary technique is the incorporation of contextual information, which adds linguistically-motivated features to the string distance module and suggests [20] that average precision in degraded texts can be reduced to a few percent.

More recent works interpret spelling correction as a statistical question, where the misspelled query is viewed as a probabilistic corruption of a correct one [2]. This approach, known as the *noisy channel model* [8], also provides ways of incorporating word pronunciation information for spelling correction in order to improve performance through the capture of pronunciation similarities between words [24]. It can also be extended to learning spelling correction models based on query re-formulations in search engine logs [4, 5].

However, whatever the concrete spelling correction technique applied, a common objection to these robust querying architectures concerns [13] the difficulty of interpreting practical results. Indeed, regardless of the location of the misspelling, retrieval effectiveness can be affected by many factors, such as detection rates of indexing features or systematic typo errors. It can be also affected by the simulation process, by the behavior of the concrete retrieval function, or by collection characteristics such as the length of documents and queries.

As a possible alternative to deal with corrupted queries in Spanish, we propose in this work a character n-gram-based strategy, since we are confident that it can avoid a number of the above-mentioned drawbacks. More exactly, our main goal is to design a robust technique adapted to efficiently analyzing short queries on which the flexibility of the IR system does not impose relevant linguistic constraints. In other words, we are interested in a methodology that is simple and ready for use independently of the documentary database considered and the linguistic resources available.

This paper is structured as follows. Firstly, Sect. 2 briefly introduces our n-gram-based proposal. Next, Sect. 3 presents the two spelling correction approaches used for comparing our proposal. Sect. 4 introduces our evaluation methodology and the experiments performed. Finally, Sect. 5 contains our conclusions and proposals for future work.

2. TEXT RETRIEVAL THROUGH CHARACTER N-GRAMS

Formally, an *n-gram* is a sub-sequence of n items from a given sequence. So, for example, we can split the word `"potato"` into four overlapping character 3-grams: `-pot-`, `-ota-`, `-tat-` and `-ato-`. This simple concept has recently been rediscovered for indexing documents by the *Johns Hopkins University Applied Physics Lab* (JHU/APL) [11], and we recover it now for our proposal.

In dealing with monolingual IR, adaptation is simple since both queries and documents are simply tokenized into overlapping n-grams instead of words. The resulting n-grams are then processed as usual by the retrieval engine. Their interest springs from the possibilities they may offer, particularly in the case of languages other than English, for providing a surrogate means of normalizing word forms and allowing languages of very different natures to be managed without further processing. Also, this knowledge-light approach does not rely on language-specific processing, and can even be used when linguistic information and resources are scarce or unavailable.

This seems to be a promising starting point from which to introduce an effective indexing/recovering strategy to deal with degraded queries. Indeed, the use of indexes based on n-grams nips in the bud the main factor justifying the integration of spelling correction methods in robust IR applications, namely that classic text recovery strategies assume exact matching on entire and correct word indexes, which are usually normalized. So, by using n-grams instead of entire words, matching should only be applied on substrings of these. In practice, this eliminates both the impact of misspelling, to which no specific attention should be paid, and the need to apply normalization. More generally, it should also greatly reduce the inability to handle out-of-vocabulary words.

3. SPELLING CORRECTION

In order to justify the practical interest of our robust IR proposal based on character n-grams, we also introduce a classic approach associated to a contextual spelling corrector [18], which will enable us to define a comparative testing frame. To begin with, we apply a global finite-state error repair algorithm proposed by Savary [21]. This technique is based on a previous one due to Oflazer [17], which searches for all possible corrections of a misspelt word that are within a given edit distance threshold. The main contribution of Savary lies in giving only the nearest-neighbors, that is, the valid repaired words with the minimal edit distance from the input. In this way, the list of correction candidates should be shorter because only the closest alternatives are taken into account, which should reduce both the practical complexity and the chance of choosing a wrong correction.

We now give a brief description about how the Savary's algorithm works. Assuming that the kernel of the spelling correction module is a *finite automaton* (FA), $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{Q}_f)$, recognizing the lexicon of the language, and where: $\mathcal{Q}$ is the set of states, Σ the set of input symbols, δ is a function of $\mathcal{Q} \times \Sigma$ into $2^{\mathcal{Q}}$ defining the transitions of the automaton, q_0 the initial state and $\mathcal{Q}_f$ the set of final states.

The procedure starts like a standard recognizer, trying to go from the initial state to a final one through transitions labeled with input string characters. When an error is detected in a word, the recognizer reaches a state from which there is no transition for the next character in the input. In that situation, four kinds of elementary *repair hypothesis* —each one corresponding to an elementary edit operation— are applied in order to obtain a new configuration from which the standard recognition may continue, namely:

- *Insertion*: skip the current character in the input string and try to continue from the current state.
- *Deletion*: try to continue from each state accessible from the current one.
- *Replacement*: skip the current character in the input string and try to continue from each state accessible from the current one. It is equivalent to applying a deletion followed by an insertion, or vice-versa.
- *Transposition*: only applicable when it is possible to get a state q from the current one with the next character in the input string, and it is also possible to get a

new state p with the current character. If those conditions are satisfied then the algorithm tries to continue from state p and skips the next two characters in the input.

to which the programmer can associate a pre-defined weight. Given that the error can be multiple or/and precipitated for a previous wrong recovery, these operations must be possibly applied recursively until a correct configuration is achieved, from both the point where the error is detected and all previous configurations of the FA. The algorithm also reduces the search space dynamically, retaining only the minimal corrections and attempting to reach the first one as soon as possible.

Unfortunately, as a result of this correction process, the algorithm can return several repair candidates that, from a morphological point of view, have a similar quality, i.e. when several words exist at the same closest edit distance from the original misspelt word.

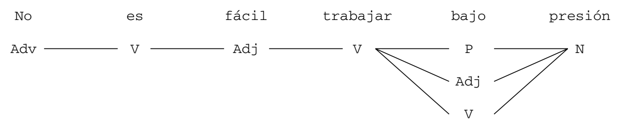


Figure 1: An example of a trellis

However, it is possible to go beyond Savary's proposal by taking advantage of the contextual linguistic information embedded in a tagging process in order to rank these candidates. We then talk about *contextual spelling correction*, whose kernel, in our case, is a stochastic part-of-speech tagger based on a dynamic extension of the Viterbi's algorithm over second order *Hidden Markov Models* [7]. The classic version of the Viterbi's algorithm [26] is applied on trellises (see Fig. 1), where the first row contains the words of the sentence to be tagged, and the possible tags appear in columns below the words, the goal being to compute the most probable sequence of tags for the input sentence. However, given that words are in nodes, it is not possible to represent different spelling correction alternatives in a trellis, since for a single position of the sentence containing a misspelling, several candidate corrected words may exist, each one with its corresponding possible tags. For this reason, we have chosen to use an extension of the original Viterbi's algorithm [7], which is applied on lattices instead of trellises (see Fig. 2). Lattices are much more flexible than trellises because words are represented in arcs instead of nodes. In the context of spelling correction, it allows us to represent a pair *word/tag* in each arc and then, by mean of an adaptation of the Viterbi algorithm equations, the probability of each possible path can be computed.

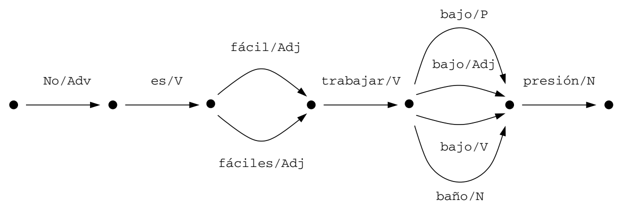


Figure 2: Spelling correction alternatives represented on a lattice

To illustrate the process with an example, let us consider the sentence *"No es fácile trabajar baio presión"*, which is intended to be a corrupted interpretation of the phrase *"No es fácil trabajar bajo presión"* (`"It is not easy to work under pressure"`), where the words *"fácile"* and *"baio"* are misspellings.

Let us now assume that our spelling corrector provides both *"fácil"/Adj-singular* (`"easy"`) and *"fáciles"/Adj-plural* (`"easy"`) as possible corrections for *"fácile"*. Let us also assume that words *"bajo"/Adj* (`"short"`), *"bajo"/Preposition* (`"under"`), *"bajo"/Verb* (`"I bring down"`) and *"baño"/Noun* (`"bath"`) are corrections for *"baio"*. We can then consider the lattice in Fig. 2 as a pseudo-parse representation including all these alternatives for correction. The execution of the dynamic Viterbi's algorithm over it provides us both with the tags of the words and the most probable spelling corrections in the context of this concrete sentence. This allows us to propose a ranked list of correction candidates on the basis of the computed probability for each path in the lattice.

4. EVALUATING OUR PROPOSAL

Our approach has initially been tested for Spanish. This language can be considered a representative example since it shows a great variety of morphological processes, making it a hard language for spelling correction [25]. The most outstanding features are to be found in verbs, with a highly complex conjugation paradigm, including nine simple tenses and nine compound tenses, all of which have six different persons. If we add the present imperative with two forms, the infinitive, the compound infinitive, the gerund, the compound gerund, and the participle with four forms, then 118 inflected forms are possible for each verb. In addition, irregularities are present in both stems and endings. So, very common verbs such as *"hacer"* (`"to do"`) have up to seven different stems: *"hac-er"*, *"hag-o"*, *"hic-e"*, *"har-é"*, *"hiz-o"*, *"haz"*, *"hech-o"*. Approximately 30% of Spanish verbs are irregular, and can be grouped around 38 different models. Verbs also include enclitic pronouns producing changes in the stem due to the presence of accents: *"da"* (`"give"`), *"dame"* (`"give me"`), *dámelo* (`"give it to me"`). There are some highly irregular verbs that cannot be classified in any irregular model, such as *"ir"* (`"to go"`) or *"ser"* (`"to be"`); and others include gaps in which some forms are missing or simply not used. For instance, meteorological verbs such as *"nevar"* (`"to snow"`) are conjugated only in third person singular. Finally, verbs can present duplicate past participles, like *"impreso"* and *"imprimido"* (`"printed"`).

This complexity extends to gender inflection, with words considering only one gender, such as *"hombre"* (`"man"`) and *"mujer"* (`"woman"`), and words with the same form for both genders, such as *"azul"* (`"blue"`). In relation to words with separate forms for masculine and feminine, we have a lot of models such as: *"autor/autora"* (`"author/authoress"`); *"jefe/jefa"* (`"boss"`) or *"actor/actriz"* (`"actor/actress"`). We have considered 20 variation groups for gender. We can also refer to number inflection, with words presenting only the singular form, such as *"estrés"* (`"stress"`), and others where only the plural form is correct, such as *"matemáticas"* (`"mathematics"`). The construction of different forms does not involve as many variants as in the case of gender, but we can also consider a certain number of models: *"rojo/rojos"* (`"red"`) or *"luz/luces"* (`"light(s)"`), for example. We have considered 10 variation groups for number.

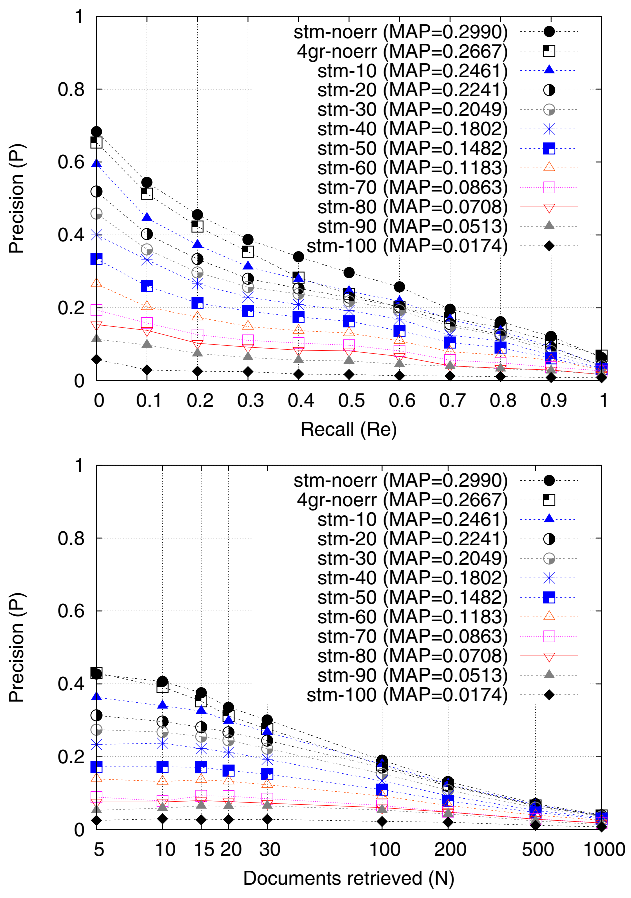


Figure 3: Results for misspelled (non-corrected) topics using stemming-based retrieval: Precision vs. Recall (top) and Precision at N documents retrieved (bottom) graphs.

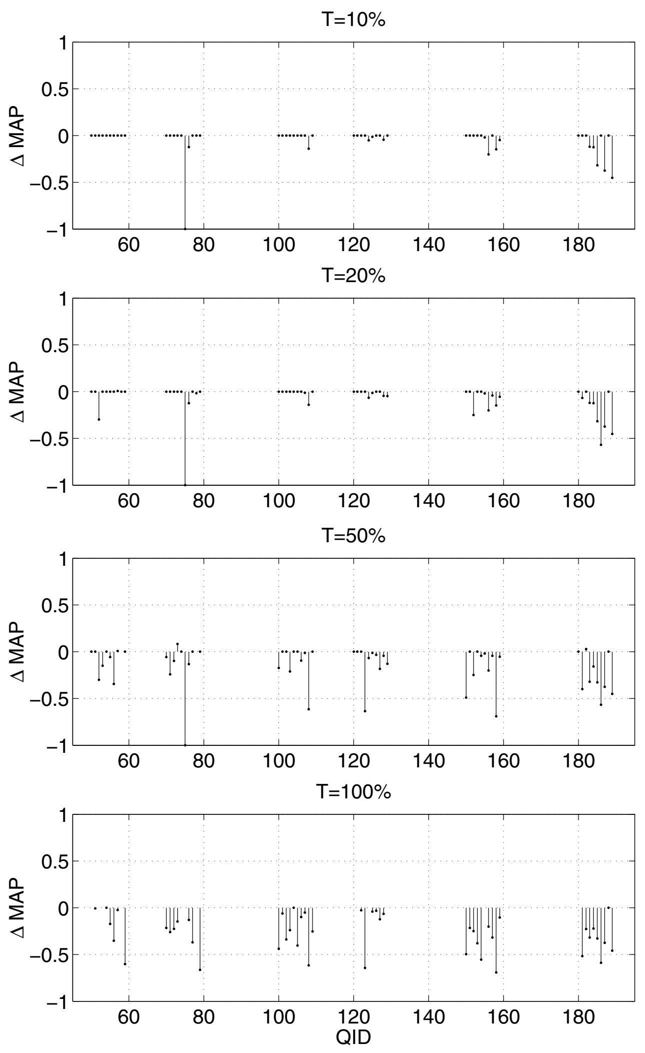


Figure 4: Per query MAP differences: misspelled (non-corrected) stemmed topics vs. original stemmed topics

4.1 Error Processing

The first phase of the evaluation process consists of introducing spelling errors in the test topic set. These errors were randomly introduced by an automatic error-generator according to a given error rate. Firstly, a *master error file* is generated as explained below. For each topic word with a length of more than 3 characters, one of the four edit errors described by Damerau [6] is introduced in a random position of the word. This way, introduced errors are similar to those that a human writer or an OCR device could make. At the same time, a random value between 0 and 100 is generated. Such a value represents the probability of not containing a spelling error. This way we obtain a master error file containing, for each word, its corresponding misspelled form, and a probability value.

All these data make it possible to easily generate different test sets for different error rates, allowing us to evaluate the impact of this variable on the output results. Such a procedure consists of reading the master error file and selecting, for each word, the original form in the event of its probability being higher than the fixed error rate, or than the misspelled form in the other case. So, given an error rate T, only T% of the words of the topics should contain an error. An interesting feature of this solution is that the errors are incremental, since the misspelled forms which are present for a given error rate continue to be present for a higher error rate, thereby avoiding any distortion in the results.

The next step consists of processing these misspelled topics and submitting them to the IR system. In the case of our n-gram-based approach no extra resources are needed, since the only processing consists of splitting them into n-grams. However, for correction-based approaches, a lexicon is needed, and in the particular case of our contextual corrector, a manually disambiguated training corpus is also needed for training the tagger. We have chosen to work with the MULTEX-JOC Spanish corpus and its associated lexicon. The MULTEX-JOC corpus [27] is a part of the corpus developed within the MULTEXT project[2] financed by the *European Commission*. This part contains raw, tagged and aligned data from the *Written Questions and Answers* of the *Official Journal of the European Community*. The corpus contains approximately 1 million words per language: En-

[2]`http://www.lpl.univ-aix.fr/projects/multext`

glish, French, German, Italian and Spanish. About 200,000 words per language were grammatically tagged and manually checked for English, French, Italian and Spanish. Regarding the lexicon of the Spanish corpus, it contains 15,548 words that, once compiled, build an automaton of 55,579 states connected by 70,002 transitions.

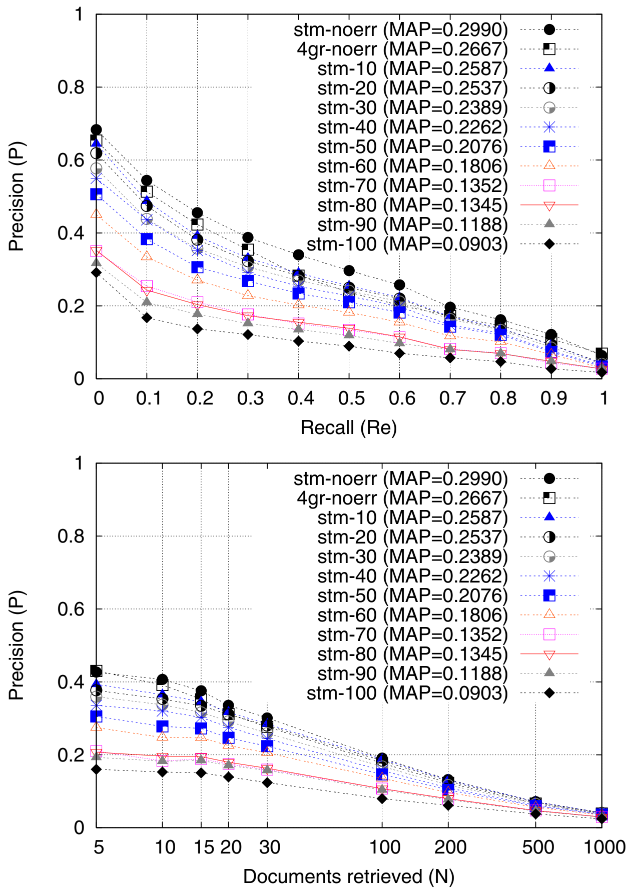


Figure 5: Results for the topics corrected through Savary's correction approach using stemming-based retrieval: Precision vs. Recall (top) and Precision at N documents retrieved (bottom) graphs.

4.2 The Evaluation Framework

The open-source TERRIER platform [23] has been employed as the retrieval engine of our system, using an InL2[3] ranking model [1]. With regard to the document collection used in the evaluation process, we have chosen to work with the Spanish corpus of the CLEF 2006 robust task [16],[4] which is formed by 454,045 news reports (1.06 GB). More in detail, the test set consists of the 60 training topics established for that task: C050–C059, C070–C079, C100–C109, C120–C129, C150–159 and C180–189. Topics are formed by three fields: a brief *title* statement, a one-sentence *description*, and a more complex *narrative* specifying the relevance assessment criteria. Nevertheless, only the *title* field has been used in order to simulate the case of short queries such as those used in commercial engines. Taking this document collection as input, two different indexes are then generated.

In order to test the correction-based proposal, a classical stemming-based approach is used for both indexing and retrieval. We have chosen to work with SNOWBALL stemmer,[5] based on Porter's algorithm [19], while the stop-word list used was that one provided by the University of Neuchatel.[6] Both approaches are commonly used by the IR research community. Following Mittendorfer *et al.* [14, 15], a second list of so-named meta-stop-words has also been used in the case of queries. Such stop-words correspond to meta-level content, i.e. those expressions corresponding to query formulation but without giving any useful information for the search. This is the case, for example, of the phrase:

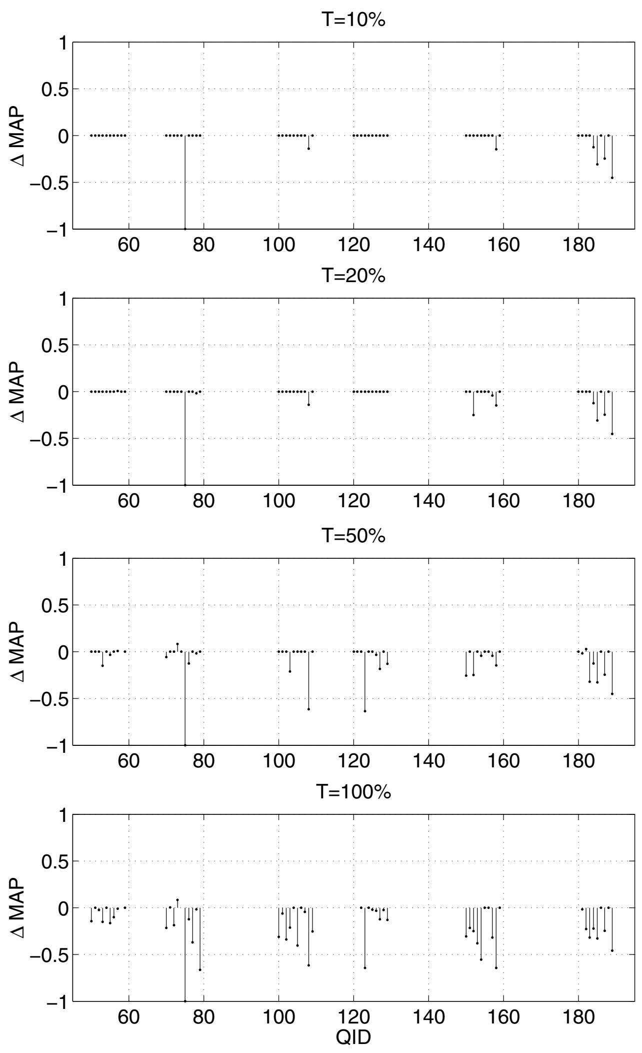


Figure 6: Per query MAP differences: misspelled corrected stemmed topics using Savary's approach vs. original stemmed topics

[3]Inverse Document Frequency model with Laplace after-effect and normalization 2.

[4]These experiments must be considered as unofficial experiments, since the results obtained have not been checked by the CLEF organization.

[5]`http://snowball.tartarus.org`

[6]`http://www.unine.ch/info/clef/`

"encuentre aquellos documentos que describan ..." (`"find those documents describing ..."`).

On the other hand, for testing our n-gram-based approach, documents are lowercased, and punctuation marks, but not diacritics, are removed. The resulting text is split and indexed using 4-grams, as a compromise on the n-gram size after studying the previous results of the JHU/APL group [12]. No stop-word removal is applied in this case.

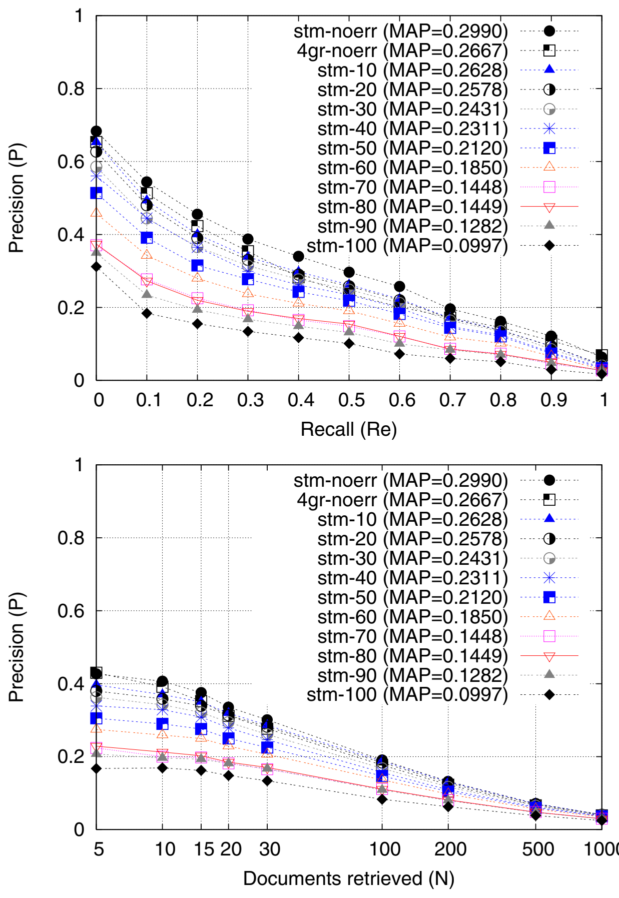


Figure 7: Results for the topics corrected through our contextual correction approach using stemming-based retrieval: Precision vs. Recall (top) and Precision at N documents retrieved (bottom) graphs.

4.3 Experimental results

Our proposal has been tested for a wide range of error rates, T, in order to study the behavior of the system not only for low error densities, but also for high error rates existing in noisy and very noisy environments like those where the input are obtained from mobile devices or based on handwriting (i.e., tablet computing):

$$T \in \{0\%, 10\%, 20\%, 30\%, \ldots, 100\%\}$$

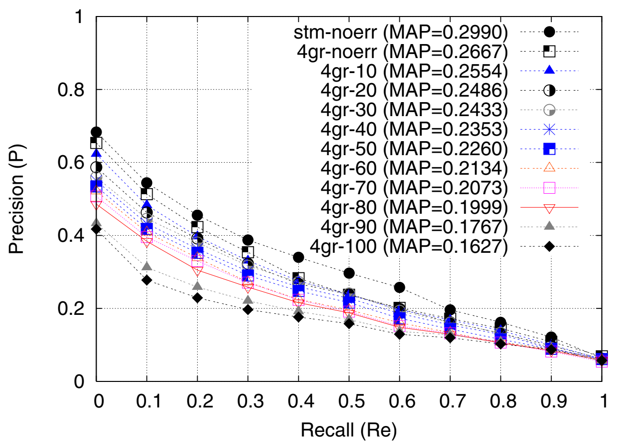
where T=0% means no extra errors have been introduced.

The first set of experiments performed were those using the misspelled (non-corrected) topics in the case of a classical stemming-based approach. The results obtained for each error rate T are shown in the graphs of Fig. 3 taking as baselines both the results for the original topics —i.e., for T=0%— (*stm-noerr*), and those obtained for such original topics but when using our n-gram based approach (*4gr-noerr*). Notice that *mean average precision* (MAP) values are also given. This first results show stemming to be sensitive to misspellings. As can be seen, even a low error rate such as T=10% has a significant impact on performance —MAP decreases by 18%—, which increases as the number of errors introduced grows: 25% loss for T=20%, 50% for T=50% (with 2 queries no longer retrieving documents) and 94% for T=100% (13 queries no longer retrieving documents), for example. Such variations, at query level, are shown in Fig. 4. All this is due to the fact that with the kind of queries like those we are using here —4 words on average—, each single word is of key importance, since the information lost when one term does not match because of a misspelling cannot be recovered from any other term.

Our second round of experiments tested the behavior of the system when using the first of the correction approaches considered in this work, that is, when submitting the misspelled topics once they have been processed using Savary's algorithm. The correction module takes as input the misspelled topic, obtaining as output a corrected version where each misspelled word has been replaced by the closest term in the lexicon, according to its edit distance. In the event of a tie —more than one candidate word at the same closest edit distance—, the query is expanded with all corrections. For example, taking as input the sample sentence previously considered in Sect. 3, *"No es fácile trabajar baio presión"*, the output returned would be *"No es fácil fáciles trabajar bajo baño presión"*. On analysis, the results obtained, shown in Fig. 5, indicate that correction has a clear positive effect on performance, greatly diminishing —although not eliminating— the impact of misspellings, not only for low error rates (MAP losses diminish from 18% to 13% for T=10% and from 25% to 15% for T=20%), but even for high-very high error rates (from 50% to 31% for T=50% and from 94% to 70% for T=100%), as well as reducing the number of queries not retrieving documents (now only 1 for T=50% and 5 for T=100%). Query level MAP differences are presented in Fig. 6. Data analyses also show that the relative effectiveness of correction increases at the same time as the error rate.

In order to try to remove noise introduced by ties when using Savary's approach, a third set of tests has been performed using our contextual spelling corrector instead of Savary's original proposal. These results are shown in Fig. 7 and, as expected, results consistently improve with respect to the original approach, although little improvement is attained through this extra processing (an extra 2% MAP loss recovery for 10%$\leq T \leq$60%), except for very-noisy environments (7–10% loss recovery for T>60%).

Finally, we have tested our n-gram-based proposal. So, Fig. 8 shows the results when the misspelled (non-corrected) topics are submitted to our n-gram-based IR system. As can be seen, although stemming performs better than n-grams for the original queries, the opposite is the case in the presence of misspellings, the latter not only clearly outperforming stemming when no correction is applied, but also outperforming correction-based approaches —except for the very lowest error rates. Moreover, the robustness of this n-gram-based proposal in the presence of misspellings proves to be far superior to that of any of the previous stemming-based approaches. As an example, MAP losses for simple stem-



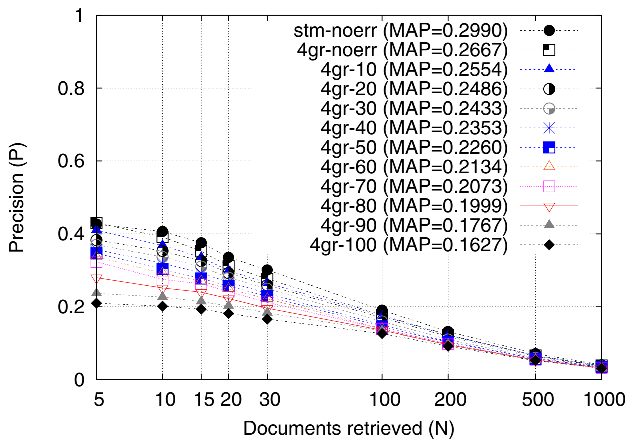


Figure 8: Results for the misspelled (non-corrected) topics using n-gram-based retrieval: Precision vs. Recall (top) and Precision at N documents retrieved (bottom) graphs.

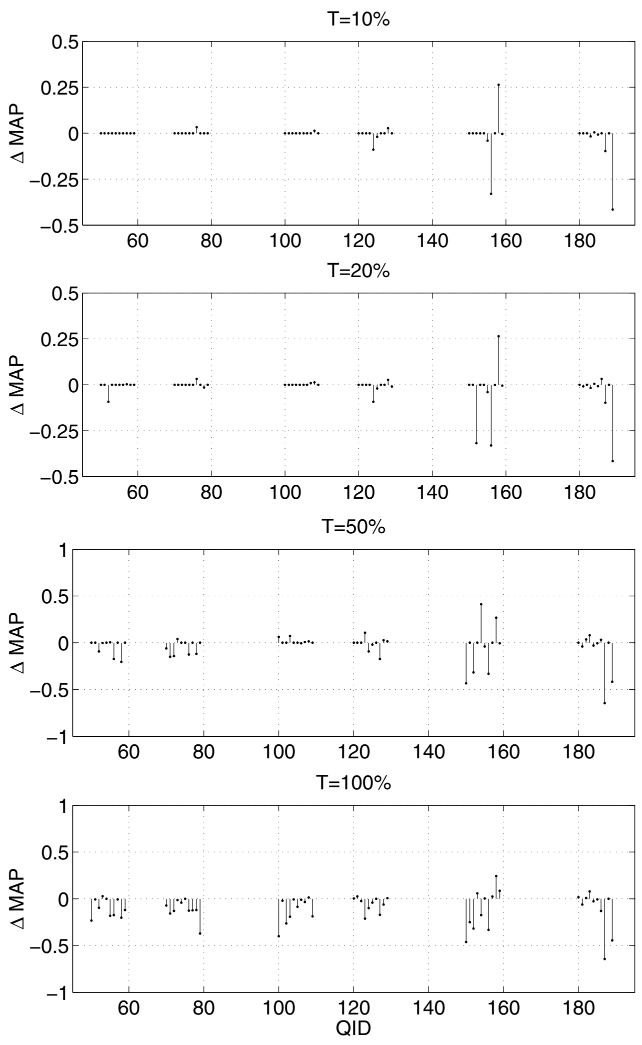


Figure 9: Per query MAP differences: misspelled (non-corrected) n-gram-based topics vs. original n-gram-based topics

ming —as stated before— were by 18% for T=10%, 25% for T=20%, 50% for T=50% and 94% for T=100%; for the same T values, the application of our contextual spelling corrector —which was slightly superior to Savary's proposal— reduced such losses to 12%, 14%, 29% and 67%, respectively; however, n-grams outperform both significantly, nearly halving these latter losses: 4%, 7%, 15% and 39%, respectively. Furthermore, there are no queries not retrieving documents, even for T=100%. Query level performance is shown in Fig. 9.

5. CONCLUSIONS AND FUTURE WORK

Our work is a first step in the design of querying techniques intended to be used in a generic, non specialized, linguistic domain of application. So, we try to favor interaction with the user through an efficient treatment of degraded queries in Spanish, avoiding classic spelling correction methods that require complex implementation, not only from the computational point of view but also from the linguistic one.

In this sense, two different approaches are proposed here. Firstly, a contextual spelling corrector is introduced as a development of a previous global correction technique but extended to include contextual information obtained through part-of-speech tagging. Our second proposal consists of working directly with the misspelled topics, but using a character n-gram-based IR system instead of a classical stemming-based one.

Tests have shown that classic stemming-based approaches are very sensitive to spelling errors, although the use of correction mechanisms allows the negative impact of misspellings on system performance to be reduced. On the other hand, character n-grams have proved to be highly robust, clearly outperforming correction-based techniques, particularly at medium or higher error rates. Moreover, since it does not rely on language-specific processing, our n-gram-based approach can be used with languages of very different natures even when linguistic information and resources are scarce or unavailable.

With regard to future work, we intend to extend the concept of *stop-word* to the case of n-grams in order to both increase the performance of the system and reduce processing and storage resources. Moreover, in order to preserve the language-independent nature of the approach, they should be generated automatically from the input texts [10].

On the other hand, it would be interesting to test the impact of the length of the query on the results, in the case of both correction-based and n-gram-based solutions. Finally, new tests with other languages are being prepared.

6. REFERENCES

[1] G. Amati and C-J. van Rijsbergen. Probabilistic models of Information Retrieval based on measuring divergence from randomness. *ACM Transactions on Information Systems*, 20(4):357–389, 2002.

[2] Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction. In *ACL*, 2000.

[3] Cross-Language Evaluation Forum. http://www.clef-campaign.org (visited in August 2008).

[4] K. Collins-Thompson, C. Schweizer, and S. Dumais. Improved string matching under noisy channel conditions. In *Proc. of the 10th Int. Conf. on Information and Knowledge Management*, pages 357–364, 2001.

[5] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proc. of Empirical Methods in Natural Language Processing*, 2004.

[6] F. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3), March 1964.

[7] J. Graña, M.A. Alonso, and M. Vilares. A common solution for tokenization and part-of-speech tagging: One-pass Viterbi algorithm vs. iterative approaches. In *Text, Speech and Dialogue*. Springer-Verlag, 2002.

[8] Mark D. Kernighan, Kenneth Ward Church, and William A. Gale. A spelling correction program based on a noisy channel model. In *COLING*, pages 205–210, 1990.

[9] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Doklandy*, 6:707–710, 1966.

[10] R.T.W. Lo, B. He, and I. Ounis. Automatically building a stopword list for an information retrieval system. In *Proceedings of the 5th Dutch-Belgian Information Retrieval Workshop (DIR'05)*, Utrecht, Netherlands, 2005.

[11] P. McNamee and J. Mayfield. Character N-gram Tokenization for European Language Text Retrieval. *Information Retrieval*, 7(1-2):73–97, 2004.

[12] P. McNamee and J. Mayfield. JHU/APL experiments in tokenization and non-word translation. volume 3237 of *Lecture Notes in Computer Science*, pages 85–97. Springer-Verlag, Berlin-Heidelberg-New York, 2004.

[13] E. Mittendorf and P. Schauble. Measuring the effects of data corruption on information retrieval. In *Symposium on Document Analysis and Information Retrieval*, page XX, 1996.

[14] M. Mittendorfer and W. Winiwarter. A simple way of improving traditional IR methods by structuring queries. In *Proc. of the 2001 IEEE Int. Workshop on Natural Language Processing and Knowledge Engineering (NLPKE 2001)*, 2001.

[15] M. Mittendorfer and W. Winiwarter. Exploiting syntactic analysis of queries for information retrieval. *Data & Knowledge Engineering*, 42(3):315–325, 2002.

[16] A. Nardi, C. Peters, and J.L. Vicedo, editors. *Results of the CLEF 2006 Cross-Language System Evaluation Campaign, Working Notes of the CLEF 2006 Workshop, 20-22 September, Alicante, Spain*, 2006. Available at [3].

[17] K. Oflazer. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–89, 1996.

[18] J. Otero, J. Graña, and M. Vilares. Contextual Spelling Correction. *Lecture Notes in Computer Science*, 4739:290–296, 2007.

[19] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[20] P. Ruch. Using contextual spelling correction to improve retrieval effectiveness in degraded text collections. In *Proc. of the 19th Int. Conf. on Computational Linguistics*, pages 1–7, 2002.

[21] A. Savary. Typographical nearest-neighbor search in a finite-state lexicon and its application to spelling correction. *Lecture Notes in Computer Science*, 2494:251–260, 2001.

[22] K. Taghva, J. Borsack, and A. Condit. Results of applying probabilistic IR to OCR text. In *Proc. of the 17th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Performance Evaluation, pages 202–211, 1994.

[23] http://ir.dcs.gla.ac.uk/terrier/ (visited on August 2008).

[24] Kristina Toutanova and Robert C. Moore. Pronunciation modeling for improved spelling correction. In *ACL*, pages 144–151, 2002.

[25] M. Vilares, J. Otero, and J. Graña. On asymptotic finite-state error repair. *Lecture Notes in Computer Science*, 3246:271–272, 2004.

[26] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Information Theory*, IT-13:260–269, April 1967.

[27] J. Véronis. Multext-corpora. an annotated corpus for five european languages. CD-ROM, 1999. DISTRIBUTED BY ELRA/ELDA.

Indexing and Retrieval of a Greek Corpus

Georgios Paltoglou
University of Macedonia
Egnatias 156, 54006
Thessaloniki, Greece
gpalt@uom.gr

Michail Salampasis
Alexander Technological
Educational Institute of
Thessaloniki
P.O. BOX 141, 57400
Thessaloniki, Greece
cs1msa@it.teithe.com

Foris Lazarinis
Technological Educational
Institute of Mesolonghi
New Buildings
3020 Mesolonghi, Greece
lazarinf@teimes.gr

ABSTRACT

Greek is one of the most difficult languages to handle in Web Information Retrieval (IR) related tasks. Its difficulty stems from the fact that it is grammatically, morphologically and orthographically more complex than the *lingua franca* of IR, English. In this paper, we address a significant number of issues that originate from the Greek language. We use a number of techniques to determine the correct encoding that is used by web pages written in Greek. We test the effect of using a Greek stopword list in a realistic and controlled Web environment. We employ a character mapping scheme, in order to overcome the problem of the diversity of diacritics used in the language, such as accents and diaeresis. We utilize word distance and fuzzy similarity metrics in order to make up for the different forms that nouns, verbs and articles appear because of conjugations and inflections and additionally handle greeklish queries, a transliterated form of Greek. The conducted experiments present some effective ways to increase the accuracy in Greek IR tasks.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Greek, Stemming, Non-English Retrieval, Web Searching

1. INTRODUCTION

Web search engines, such as Google, Yahoo! or Ask, have brought Information Retrieval (IR) to the masses as they are utilized daily by millions of people. IR is the scientific field that deals with the general problem of retrieving information, usually, but not exclusively, in the form of unstructured or semi-structured textual data, in order to satisfy the user's information need.

Most of the research that is being done in the field aims toward English-speaking users. One could say that English is the *lingua franca* of the Web in general and of IR systems in particular. Standard techniques, such as stemming [11] and stop word removal were originally designed and explored for the English language.

Nonetheless, most of Web users are non-English and therefore the amount of queries that are posed to search engines, as well as the amount of information that is available on the Web in languages other than English is rapidly increasing [2]. As the number of non-English users increases, so will the efforts by the IR community and search engines to address the individual issues of popular languages. The Cross Language Evaluation Forum (CLEF) Web Track (WebCLEF[1]) is a result of the above interest, constituting an organized effort into dealing with information retrieval issues in a multilingual environment.

One of the most "difficult" languages to deal with in IR related problems is Greek. Indicatively, the results from the WebCLEF tracks for Greek queries have been one of the lowest in the track [8]. The difficulty of the Greek language results from the fact that it is grammatically, morphologically as well as syntactically significantly more complex than the English language. It utilizes conjugations for verbs and inflections for nouns, articles and verbs (nominative, genitive, accusative and vocative). Even more, articles, nouns and verbs in Greek have different singular and plural forms. Additionally, there is a significant number of irregular terms that do not follow any rules and are inflected in specific ways. Greek orthography is also by itself rather complex, making use of a number of diacritics, such as accents (i.e. ί), diaeresis (ϊ) and even combinations of both (ΐ) which are imposable on lower-case vowels. Accents are also used in upper-case vowels when they are the first letter of a word. Last but not least, when the character σ is used at the end of a word it is transformed into its variant, ς. The above have resulted in a reduced effectiveness of Greek IR tasks.

2. PRIOR WORK

The rapid increase of non-English Web users and the parallel increase of highly qualitative content in languages other than English have created a significant interest in multilingual Information Retrieval.

The WebCLEF workshops [2] present an organized ef-

[1]http://ilps.science.uva.nl/WebCLEF/

fort by the IR community to address the issues of retrieval in a multilingual setting within a common and controlled environment. A significant amount of research has been presented within the workshops, addressing information retrieval issues in this linguistically mixed environment. Standard techniques used on IR for decades, such as stemming [8] and stop word removal were extended to other languages and tested as part of the WebCLEF workshops.

The Greek language, as already noted, presents particular difficulties due to its multilevel complexity, which make Greek IR tasks even more difficult. This difficulty becomes apparent also by the lack of adaptation of standard IR techniques to Greek. Stemming, for example that has often proved quite effective in IR is still an open issue in Greek. Although stemming algorithms for languages other than English have been developed (i.e. see Snowball's site[2]) no widely adopted solution has been proposed for the Greek language. Various approaches have been presented in the past [6, 13, 9] but little effort has been made to test them in IR related tasks.

Additionally, there has only been limited research in the development of a stopword list in Greek. In [5] the process of developing of a stop word list for Greek was presented and its effectiveness was tested by posing queries to a public commercial search engine (Google) with and without the use of stop words. Nonetheless, since the algorithms employed by commercial search engines are rather complex, the experimental setup gives limited insight to the isolated effect of eliminating stop words from the query. In this paper, we utilize the developed stop word list from that work both in the indexing process as well as in the queries posed to the system, in a realistic but isolated Web environment to examine the way it affects retrieval quality. Additionally, we aim to compute the gain in index size that results from the elimination of Greek stop words.

Apart from the above issues of adapting standard IR techniques to the Greek language, little effort has been given to address the particular intricacies of the language, such as encoding issues of web pages written in Greek, its complex orthography, the utilization of diacritics within words and the different forms that terms with identical grammatical roots and similar semantic meaning are encountered due to conjugation and inflection.

The study that is presented here aims to address some of the above issues. The experiments conducted will offer solutions to the encoding issues of Greek web pages. A mapping process will attempt to address the issue of diacritics within Greek documents as well as provide a solution to handling queries in romanized or latinized Greek (called "greeklish", more on paragraph 3.1). Additionally, with the usage of word distances and fuzzy similarity measures we will attempt to elevate the problem of conjugation and inflection, expanding the query to include variations of the original query terms.

3. INDEXING AND RETRIEVAL OF GREEK TEXT

3.1 Pre-processing and indexing

One of the key issues regarding the indexing and retrieval of Greek text on the Web is encoding. Presently, the most frequently used encoding schemes that support the Greek language are: WINDOWS-1253, ISO-8859-7 and UTF-8. The actual encoding of a web page depends on a variety of factors, such as the platform and the tools that were used during development, the expertise of the web developer as well as the intended audience. Although, most of government webpages are written in ISO-8859-7, the problem of different encoding schemes is very existent and significant in the rest of the Greek domain. For this reason, after determining the actual encoding of the webpage, we convert it into UTF-8. The actual determination can be done either through the page's META tag, by examining the "charset" property or alternatively using heuristic rules, such as looking for Greek characters using the above schemes and choosing the one that produces Greek characters.

Having encoded the pages at a readable form, a "character mapping" process takes place. The mapping process aims to reduce all Greek characters whether upper or lower-case to a controlled set of predetermined characters. For example, the above process maps the characters ι and I to the same form *I*. Additionally, the mapping was designed in such a way as to ignore diacritics, therefore additionally mapping the forms ί , ϊ and ΐ as well as their capital variants Ί and Ϊ all to the same character *I*. The above process has the advantage of removing a significant part of the intricacies of the Greek orthography and additionally helps with potential user spelling errors, caused by the misuse of diacritics. On the other hand, it may introduce some errors and misinterpretations of words whose difference is based solely on the use of diacritics. For example, both words παιδάκια (little children) and παϊδάκια (lamb chops) are mapped to the same form ΠΑΙΔΑΚΙΑ. Despite this drawback, we believe that the number of words that are distinguished only by diacritics is rather limited and therefore the above mapping scheme was adopted. We leave the design of a more complex mapping process that takes under consideration diacritics when there is a potential conflict for future work.

The controlled character set that would be used was also open for discussion. In the Greek language there are a lot of vowels and diphthongs that produce the same sound. For example, the characters ι, η, υ as well as the diphthongs ει and οι all produce the same sound [i]. In order to keep the original text to a form as close to the original, it was decided to follow a controlled set of twenty-four letter, the same number as the Greek alphabet. Nonetheless, one could opt to utilize less than the original number, for example mapping all the above vowels to the same letter I, in a way adopting a "phonetic" mapping scheme. Such a mapping scheme would indirectly provide an spell checking functionality, for example mapping the correctly spelled word κυβέρνηση (government) and a misspelled variation, such as κυβέρνιση to the same form ΚΙΒΕΡΝΙΣΙ. Early experiments, nonetheless, indicated that such a mapping process would reduce a significant number of semantically and grammatically different words to the same final form (i.e. mapping both terms τοίχοι (walls) and τύχη (luck) to the same ΤΙΧΙ), therefore introducing a great deal of noise to the retrieval process and deteriorating retrieval quality.

There was also the possibility of using a controlled set of non-Greek characters, such as English. The English alphabet has twenty-six letters, so they are more than sufficient to handle the Greek character set. Using the English alphabet, the above word ΠΑΙΔΑΚΙΑ would be mapped to

[2]http://snowball.tartarus.org/

"PAIDAKIA". The difference is trivial, under the condition that the same mapping would also be applied to all queries posed to the IR system, thus mapping all Greek characters to the same set of English characters, but there is a significant advantage for queries posed in greeklish.

Greeklish is a transliterated form of Greek using English characters and was originally used in order to make up for the lack of support of Greek characters in computer systems in the 1990's [14]. Nowadays, most systems fully support the Greek alphabet, but there is still an important number of users, especially the more technologically-savvy that prefer the use of greeklish, especially in e-mail systems, in order to avoid encoding issues. An IR system that uses English characters to map the original Greek characters has the additional advantage of being able to handle queries posed in greeklish. To make this more clear, the system would return the same results to the user posing a query in Greek, for example βασεις δεδομενων (databases) and its greeklish variant "baseis dedomenwn", since the mapping of the former would result in the latter. Experiments with users [14] have indicated that a large percentage of Greek search engine users (67.5%) would like to read Greek pages returned for queries posed in greeklish. The above mapping scheme greatly facilitates this desired feature. Table 1 presents the mapping that is eventually used, taking into consideration all the above factors.

A point worth making is that because the rules of transliterating Greek characters using English characters are not defined (at least in the minds of Greek users), the above mapping would only be able to handle greeklish queries to a certain extent. For example, the above query βασεις δεδομενων may also be transliterated as "baseis dedomenon", which is different than the one produced above, using a different set of rules. We make up for this inefficiency by expanding the original query with additional terms that are within a certain distance (see paragraph 3.5) from the original query terms.

Original Character	Mapped Character	Original Character	Mapped Character
α, ά, Α, Ά,	a	ν, Ν	n
β, Β	b	ξ, Ξ	j
γ, Γ	g	ο, ό, Ο, Ό	o
δ, Δ	d	π, Π	p
ε, έ, Ε, Έ	e	ρ, Ρ	r
ζ, Ζ	z	σ, ς, Σ	s
η, ή, Η, Ή	h	τ, Τ	t
ϑ, Θ	u	υ, ύ, ϋ, ΰ, Υ, Ύ, Ϋ	y
ι, ί, ϊ, ΐ, Ι, Ί, Ϊ	i	φ, Φ	f
κ, Κ	k	χ, Χ	x
λ, Λ	l	ψ, Ψ	y
μ, Μ	m	ω, ώ, Ω, Ώ	w

Table 1: Character Mapping rules

It must be noted here that at a realistic Web environment, it is considered crucial to locally cache two versions of the Greek text. The first version will undergo the above mapping process and will be subsequently used for indexing and retrieval, while the second version will be the original Greek text that will be returned to the users in response to their queries. After a user poses a query to the system, the retrieval mechanism will rank the indexed documents and will return to the user their original cached versions, as to facilitate the result browsing process.

3.2 Removal of stopwords

In [6] a stopword list for Greek Web retrieval was developed. That work also reports that the results that are returned from Google when stopwords are eliminated in Greek queries, are usually of higher quality and less in volume. The papers concludes that elimination of words during indexing will most likely positively affect the efficiency of the search engine, reducing the time it takes in order to respond to user queries, as well as increase the relevance of the results. The experiments conducted nonetheless are limited since they only report on the usage or elimination of stopwords only during the querying stage and not during the indexing phase.

Extending the work done in that paper, we use the produced stopword list in the indexing phase and examine whether the elimination of very frequent terms, significantly reduces the size of the index, as is expected, but most importantly whether their elimination increases retrieval effectiveness in a significant manner.

Although the removal of stopwords has been standard practice in IR, lately it has been reported that commercial search engines have stopped removing stop words and index them normally. For example, the query "what is information retrieval" returns different documents from "information retrieval", which shouldn't be the case if the stopwords "what" and "is" were removed as stopwords. The assumption behind this decision possibly results from the fact that stopwords are encountered very frequently in documents, therefore they have a very limited discriminating power which translates into a small *idf* value, eventually not affecting the final results in a significant manner. Additionally, stopwords are important in phrase search, when they appear within the requested phrase. Here, we aim to investigate the effect of removing very frequent Greek terms from the index of a search engine.

3.3 Retrieval

All the experiments reported in this paper are done using the Lemur Toolkit[3], first presented in [10], which is an information retrieval toolkit originally designed for language modeling but used extensively in the IR academia. We use the Inquery search engine, provided by the toolkit in order to rank and retrieve documents. The belief function used by Inquery to estimate the belief of query term t within document d is:

$$w_{t,d} = 0.4 + 0.6 * \frac{tf_{t,d}}{tf_{t,d} + 0.5 + 1.5 * \frac{length(d)}{avg\_len}} * \frac{log\frac{N+0.5}{n_t}}{logN + 1} \quad (1)$$

where n_t is the number of documents containing term t, N is the number of documents in the collection, avg_len is the average number of words in documents in the collection, $length(d)$ is the number of words in document d and $tf_{t,d}$ is the number of times term t occurs in document d. Lastly, 0.4 is the "default belief", set to the default value of the system. The first fraction of equation 1 estimates the significance of term t in document d (i.e. how descriptive is the term of the content of the document) and is often refered to as the

[3]http://www.lemurproject.org

tf component (in this case, the "Okapi tf" function) and the second fraction estimates the discriminating power of the term in regard to the whole corpus and is often referred to as the *idf* component. For more information see [1].

3.4 Wildcard usage in query formulation

A potential but unexplored way to discover semantically similar but grammatically different variations of Greek terms is with the usage of wildcard queries. The use of wildcards may have some merit, because if the wild card character (*) is placed at the optimal position within a term it has the potential of significantly aiding retrieval, by retrieving documents that contain not only the original query term but its inflective products. For example, the placement of the wildcard after the last τ in the word μαθητής (student), effectively transforms the query term into μαθητ* and results in the IR system returning documents also containing the terms: μαθητών, μαθητές, μαθητευόμενος, μαθήτριες etc., all which are derivatives of the original query term. Special attention must be given to the position where the wildcard is placed. Placing the wildcard too much toward the beginning of the word would most likely result in a number of unrelated terms appearing. For example, placing the wildcard character at the same word μαθητής just one position to the left, after the η (μαθη*), results in documents containing μαθηματικά (mathematics), μαθημένος (learned) returned, which are semantically different from the original query term.

Therefore, the use of wildcards could help the process of finding semantically similar terms, possibly overcoming the problem of inflections in Greek documents, but overuse will result in a significant topic drift, returning documents unrelated with the initial query term.

In the line of experiments that were conducted we tested the utilization of wildcards with a number of parameters. Early experiments indicated that using wildcards in small words (less than 4 characters in length) resulted in a serious degradation of precision due to the significant amount of irrelevant terms that were added to the original query. Therefore, we limited the application of wildcards only to words that are more than 4 characters in length.

We introduce the wildcard character after the middle letter, both on consonants and vowels to see whether there is a "sweet spot" for automatically placing the character to user queries. The aim of such a system wouldn't be to force users to use wildcards but rather to make use of them "behind the scenes", in order to increase the relevance of the returned results. In effect, the IR system will use wildcards in order to expand the initial user query and return documents that contain not only the original terms but also their widcard variations. Of course, the findings are very much related to the actual length of the query terms (a longer query term will have more places to place the wildcard character than a shorter query term), but the aim of the experiments is to investigate in general whether adding wildcards to Greek queries has any merit or not.

3.5 Query expansion using Levenshtein Word Distance

The Levenshtein distance [7] is a metric for measuring the amount of difference between two sequences (also known as edit distance). The distance between two sequences of characters (such as words) is defined as the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. The utilization of the metric allows for the discovery of words that are close morphologically and potentially semantically, to the original query terms. For example, the Levenshtein distance between μαθητής (student) and μαθήτρια (female student), not considering accents, is three: replacement of the η after τ with ρ, replacement of the ς with ι and finally insertion of α at the end of the word.

The advantages of using a distance metric are numerous. First of all, it allows for the discovery of grammatically similar terms (potentially inflations of query terms), which are also to some extend expected to be semantically similar. Secondly, it allows for spell-checking operations to take place before retrieving any documents. Last, but not least, it facilitates retrieval with greeklish queries. Taking into consideration that the mapping of the Greek characters is done using english characters, the above process allows for the discovery of words which are similar to the original Greek, but transliterated with different rules. For example, the original query term μαθητής may be transliterated in greeklish as: "mathitis", "ma8hths" or "ma8itis" all of which are within a distance of 3 (3, 1 and 3 respectively) from the actual mapping of the word (which is "mauhths"). Therefore, by extending the original greeklish query with terms that are found within the index, the system is able to make up for the variations in greeklish transliteration, returning results that contain not only the original greeklish term, but also a number of variations, as well as documents written with Greek characters.

The incorporation of the Levenshtein distance into the retrieval process is done as follows. Given a user query, the IR systems first locates all words in the corpus index that are within a certain distance and then adds the discovered terms into the original query. Then, it uses the expanded query to search the index. Expanding the query with words that are very close to the original terms, for example at a distance of 1 or 2, may avoid the problem of adding unrelated words to the query, but may also miss on relevant terms. On the other hand, setting a distance threshold too high, will most likely include all words that are related to the original terms, but will also most likely introduce a great deal of unrelated words. We experimented with various distance thresholds to find the one that produces the best results.

3.6 Query expansion using fuzzy similarity measures

Extending the above idea, we experimented with expanding the original query with terms using a fuzzy similarity measure. "Fuzzy string search" is used to describe a category of techniques for finding strings that approximately match some given pattern string. It is also known as approximate or inexact matching. The fuzzy measure used[4] is based on the number of common n-grams between words, giving a similarity measure of 0 for words that have no common sequences of characters and 1 for identical words.

The methodology was incorporated into the retrieval process similarly to the incorporation of Levenshtein Word Distance. Given the original query term the algorithm locates words from the corpus index that are within a certain threshold and adds them to the query. Then, we query the corpus using the expanded query term set.

[4]The source code is available at: http://www.ai-search.4t.com/fuzzy_string_matching.html.

It must be noted here that locating words that are similar to the original query terms, using the corpus index is a rather inefficient process for a realistic Web environment. The aim of this line of experiments is to investigate whether the expansion of the query offers using such metrics offers any advantages for Greek information retrieval and not to test the efficiency issues. We believe that those can be elevated with a more efficient implementation of the proposed distance metrics (see for example [3]).

4. EXPERIMENT SETUP

There are inherent difficulties when conducting experiments with a language that isn't widely used, one of the more important ones being the lack of standard testbeds. Three approaches were possible and were examined before finally deciding on the actual testbed that would be used in the experiments.

Obviously, the first candidate test collection was the EuroGOV test collection [12], which has been extensively used within the WebCLEF community and was the closest to a standard test collection that could be utilized. This choice of test collection, would direct us at evaluating our system at the mixed monolingual task from WebCLEF 2005 and 2006. The tasks simulates a user searching for a known-item page in a European language. It uses known-item topics, in particular homepage finding and named page finding queries. Unfortunately, there are certain issues with the specific collection that limit its usability in Greek IR experiments. First of all, the number of queries that are available in Greek is rather limited (16 in total) and secondly and most importantly, the crawl of the Greek domain is especially shallow (EuroGOV contains webpages mainly from government sites), containing only 300 web pages, of which only 288 are unique and not duplicates, to a total compressed size of 416K, which was considered extremely small for the case of the Greek Web.

Secondly, we had the opportunity to utilize an online Greek encyclopedia[5] as a test collection. The particular test collection would offer a highly qualitative content, addressing a rich variety of subjects (science, art, history etc) therefore offering a prosperous ground for IR experiments. For queries, we had to our disposal the log file of the utilization of the search function of the site of one month's duration, about 1,000 queries. Unfortunately, the encyclopedia is under copyright restrictions, a limitation that would potentially prevent us from making comparable and verifiable experiments. In order to lift this limitation but still experiment with content of similar nature and variety, the Greek version of wikipedia[6] was used. The site was crawled, a process which created a local copy of 26,589 articles, which left us with 19,372 articles after duplicates were removed. Those were parsed in order to extract their contents and remove any HTLM markup.

The above approaches suffered from one major drawback and that was the lack of any relevance judgments. Additionally, the lack of click-through data prevented us from extracting relevance judgments in a automatic or semi automatic fashion. Thus we resorted to choosing a small subset of 20 queries from the available log file and manually create relevance judgments for them. Although the number of queries is small, it is still comparable to the number that would otherwise be available to us, should we have used the standard EuroGOV. Special attention was given so that the queries would include articles (potential stop words), as well as conjugated forms of nouns. Letters in both upper and lower case were also used. The average number of terms per query is 2.8, which is compatible with what is generally reported [4]. Time limitations prevented us from creating relevance judgments for more queries. Table 2 presents the queries.

We kept the same task as in EuroGOV, that is named-page finding. Although, we would prefer conducting experiments in more of ad-hoc style queries, previous research has indicated that those tasks closely resemple the typical behavior of Web users. We also used the same metric as in WebCLEF, namely the mean reciprocal rank (MRR). The reciprocal rank is calculated as 1 divided by the rank at which the first relevant page is found. The mean reciprocal rank is obtained by averaging the reciprocal ranks of a set of topics.

Greek Query	Translation of Query
ιστορια της Βυζαντινής μουσικής	history of Byzantine music
Ελληνική Ορθόδοξη Εκκλησια	Greek Orthodox Church
τι είναι ο αλγοριθμος	what is algorithm
αρχή της απροσδιοριστίας	principle of indetermination
μηχανές αναζήτησης στο διαδίκτυο	search engines on the Web
δημοτικά τραγουδια	folk songs
λαϊκή λατινική γλωσσα	popular latin language
πνευματική ιδιοκτησία	intellectual property
βάσεις δεδομένων	databases
Ευρωπαϊκή Ένωση	European Union
Ολυμπιακοί Αγώνες	Olympic Games
Ελληνική Μυθολογία Μοίρα	Greek Mythology Fate (also known as Moira)
Αρχαία Ελληνική Ιστορία	Ancient Greek History
Σεπτέμβριος 2007	September 2007
πολιτική και θρησκευτική προπαγάνδα	political and religious propaganda
κλάδοι της Ιατρικής	branches of Medicine
νόμοι της Φυσικής	laws of Physics
Δημοψήφισμα του 1974	Referendum of 1974
Μικρασιατική καταστροφή	Catastrophe of Asia Minor (a historical event that took place at 1921-22)
Πυρηνική Φυσική	Nuclear Physics

Table 2: Queries that were used in the experiments.

It should be noted here that we do not claim to have built a test collection *at par* with standard test collections, only that the proposed testbed will offer some insight to the utilization of the aforementioned methodologies of querying the Greek Web. There is great potential on the development of a standard Greek test collection for IR experiments, but that is beyond the scope of the current work.

All experiments were done using the Inquery search engine as described in paragraph 3.3. Although it is usually advisable to use different weights for document fields (such

[5]http://www.gnosinet.gr
[6]http://el.wikipedia.org

as *title*, *body* etc) and *anchor text* as additional evidence of ranking documents in Web oriented tasks [8], we refrained from doing so, since the focus on this work is to test the merit of the methodologies presented above as ways to improve retrieval quality in the Greek language.

5. RESULTS

5.1 Removal of stopwords

Table 3 reports the initial results obtained using the mapping scheme described in paragraph 3.1, with and without the removal of stopwords. We tested with three different settings:

- **AllWords:** No stopword list was utilized during indexing and every word was indexed normally. Additionally, all queries are posed as presented in Table 2.
- **StopWordsQueries:** Stopwords are removed from the queries. In effect only 7 (35%) of 20 queries have stopwords (10 removals in total). The terms that were removed are: τῃς, τι, είναι, ο, στο, και and του.
- **StopWordsAll:** Stopwords are removed both from the index and the queries.

AllWords	0.3901
StopWordsQueries	0.3898
StopWordsAll	0.3892

Table 3: Mean Reciprocal Rank (MRR)

As expected the index created when stopwords are removed is smaller in size. In the experiments that were conducted, using the standard .key index that the Lemur Toolkit creates, which additionally holds term position information besides simple posting lists, the index was 57.8MBs for the index without stopwords and 76.7MBs with stopwords, a significant reduction of 24.6%.

Effectiveness, on the other hand remains almost unaffected by the usage or elimination of stopwords. A slight increase was actually observed when stop words were used both during the indexing and the querying phase. One would expect that the results of the *StopWordsQueries* and *StopWordsAll* runs would be identical, but the small difference observed most likely depends on the different weights attributed to the individual query terms when stop words are eliminated in the querying phase.

Overall, it can be concluded that if efficiency is of importance and the available resources are limited then the removal of stopwords does offer a significant gain. The differences in effectiveness are too minimal to conclude whether stopword removal in Greek IR actually affects positively the final results. A larger query set with heavier usage of stopwords could potentially offer more evidence.

5.2 Wildcard Queries

Next, we experiment with the usage of wildcard queries. The wildcard character is inserted at various positions in the original query terms and all words matching the form are added to the query. Results are reported at Table 4.

Early experiement showed that placing the wildcard character before the middle of the query term resulted in a serious degradation of effectiveness, because of the introduction of a significant number of unrelated words. For example, the word ιστορία (history) was expanded with the words: ιστοσελίδα (webpage), ιστιοπλοία (sailing) etc, all of which are totally unrelated with the original term. Placing the wildcard character more towards the end of the word (middle of term plus 1,2 or 3 positions toward the end), produced more related terms and eliminated a significant number of irrelevant ones. Additionally, we also experimented with placing the wildcard character in a position relative to the total length of the query term. For example if the term had a length of 7 characters, we introduced the wildcard character at positions 3, 4, 5 and 6.

Wildcard character position	MRR	Wildcard character position	MRR
TermMiddle	0.3082	TermLength - 4	0.2487
TermMiddle + 1	0.3972	TermLength - 3	0.3315
TermMiddle + 2	0.4244	TermLength - 2	0.4056
TermMiddle + 3	0.4958	TermLength - 1	0.4301

Table 4: MRR when the original query terms are expanded using the wildcard character at various positions

The results show that placing a wildcard character too early in the query term, for example in the middle, results in a serious degradation of effectiveness of almost 21% compared to the baseline (*AllWords* run at Table 3). When the wildcard is placed more towards the end of the term, the performance increases significantly and even surpasses the baseline performance when it is placed 2 and 3 positions after the middle. The last result in particularly impressive, showing an increase of 27% in comparison to the baseline, indicating that the usage of the wildcard character indeed has some merit to it.

In an attempt to extend the utilization of wildcards even more an additional condition was examined. We imposed a limit on the length in characters of the words that would be added to the query. The motivation behind this experiment is that most users enter terms in the nominative inflection [5], which is usually the longest singular form of a word. Therefore, we added only words whose length in characters doesn't exceed a specific threshold, compared to the original term. We used the best run from table 4, adding the wildcard character 3 positions after the middle of the query term. The results are reported on table 5.

Limit in Character length	MRR
Length of Original term +1	0.4522
Length of Original term +2	0.5700
Length of Original term +3	0.5196
Length of Original term +4	0.5196

Table 5: MRR when a limit on the number of characters on the added terms is imposed. The wildcard character is placed 3 positions after the middle of the query term.

The results indicate that when the limitation is too strict (original term length plus one), a small deterioration is observed, most likely because the above threshold prevents a significant number of related but slightly longer terms to be entered in the query. The "sweet spot" seems to be the limitation of two extra characters in length, which shows a non-trivial improvement of 15% to the best so far performance (*Middle of QueryTerm + 3* from table 4).

Overall, it can be concluded that the usage of whildcard queries with limitations has significant advantages. The observed increase in result quality is indicative of the fact that a significant number of semantically similar but morphologically different words are added to the query and unrelated terms are avoided.

5.3 Using distance measures

In the experiments presented in paragraph 5.2, every word that matched the wildcard term, depending on the limitation imposed, was added to the query. In this section, we experiment with adding words that are below a similarity threshold to the original query terms. The process followed is described in detail in paragraph 3.5. Results are presented in Table 6.

Lev. Thres.	MRR using Distance only	MRR using Distance and * character in middle position	MRR using Distance and * character in middle+3
1	0.4615	0.4537	0.4862
2	0.4924	0.5170	0.5478
3	0.3953	0.4936	0.5196
4	0.2327	0.4592	0.4934

Table 6: MRR using Levenshtein Word Distances.

The initial results are reported on the second from the left column. Using a high threshold (three or four) allows for the introduction of a significant number of unrelated terms to the query. On the contrary, the results using a low threshold are above the original baseline (*AllWords* run at Table 3), but not as encouraging as one would expect. Examining the actual terms that were added to the query it was observed that all of the words that are added are morphologically very similar to the original query terms, but often not semantically. For example, for the word νόμοι (laws), the word τόμοι (tomes) was added. The result is due to the fact that although the words are of distance one, their beginnings are different, therefore they most likely have a different root. In order to make up for this deficiency, we incorporated the condition that the first half of the words must be similar, effectively combining the wildcard queries examined above (where the wildcard character is entered in the middle of the word) with a distance measure. The results are reported on the third column of table 6.

The results are greatly improved. Strong indications are beginning to appear that the optimal threshold is two, after which there is usually a degradation of the quality of the results. In order to further verify the above assumption and test the extend to which Levenshtein Word Distance measures can aid Greek IR, we placed the wildcard character at three positions to the right, after the middle of the original query terms, which produced the best results in Table 4. The results are again improved, especially for higher threshold where the degradation previously observed is contained.

In an attempt to achieve optimal performance, we tested the usage of distance measures combined with the best performance that was reported using wildcard queries (Table 5). The results are reported at Table 7.

Levenshtein Word Distance Threshold	MRR
1	0.4862
2	0.5779
3	0.5700
4	0.5700

Table 7: MRR using a combination of Levenshtein Word Distance threshold with wildcard queries. The wildcard character is placed three positions right of the middle character and additionally the words that are entered to the final query are at maximum two characters longer than the original query term.

The results demonstrate that using a combination of wildcard queries with distance measures produces the best results (MRR of 0.5779). Our initial assumption that a distance of two is optimal is verified, despite the fact that because of the extra limitation set in the last experiment, the degradation in the final results using a higher threshold is contained. Surprisingly, the results aren't significantly different from the simple usage of wildcard queries, under the assumption that the wildcard character is placed at the optimal position and a limit to the maximum length of the expanded words is set (as in table 5). The results show that although helpful, sophisticated distance measures aren't necessary in order to achieve optimal performance in Greek IR.

5.4 Experiments using a fuzzy similarity measure

Experiments showed a correlation between fuzzy similarity measures based on n-grams and the Levenshtein Word Distance utilized above. Therefore, we only present a baseline experiment using various fuzzy similarity thresholds and a combination of fuzzy similarity measures and wildcard queries with optimal settings, as reported above. Results are reported on Table 8. The advantage of using fuzzy similarity measures is that they offer a greater chance of finetuning, in comparison to Levenshtein word distance measures which are only integers.

The results indicate that fuzzy and distance similarity metrics are complementary, producing similar results for various thresholds. The best performance reported is observed using a threshold of $0.75 - 0.7$, which is only slightly better from the optimal performance reported using distance metrics or wildcard queries.

6. CONCLUSIONS AND FUTURE WORK

In this paper, an examination of using the Greek language in Information Retrieval tasks was presented. Greek has been characterized as a particularly difficult language in the field of IR. Its difficulty originates from its inherent complexity, which is morphological, orthographical, grammatical and syntactical.

We attempted to give a overview of techniques that can be utilized in order to lift the intricacies and peculiarities of

Fuzzy similarity Distance Threshold	Baseline MRR	MRR using Fuzzy similarity with wildcard queries with optimal parameters
0.9	0.3816	0.4074
0.85	0.4607	0.5096
0.8	0.4723	0.5128
0.75	0.5440	0.5792
0.7	0.5430	0.5792

Table 8: MRR using a combination of Fuzzy similarity thresholds with wildcard queries. The wildcard character is placed three positions right of the middle character and additionally the words that are entered to the final query are at maximum two characters longer than the original query term.

the Greek language. We employed a Greek-to-English character mapping scheme in order to avoid issues originating from the use of diacritics, such as accents and diaeresis in Greek words. Although the mapping, at this stage, doesn't take into consideration possible conflicts it proved to be a very effective way to handle the Greek alphabet. An additional functionality of the above used mapping is that it facilitates retrieval when the queries are posed in greeklish, a transliterated form of Greek.

We also experimented with the usage of a stopword list. The results showed that when the queries make an average use of stopwords, the effects of eliminating them, both from the queries and the index are minimal in terms of result quality. Nonetheless, if the available resources are limited, the elimination of stopwords significantly aids, resulting in a much smaller index. In a realistic environment where millions of webpages need to be indexed, the above reduction may be of importance.

One of the most difficult aspects of the Greek language is the variety of forms that words that have the same semantic meaning and morphological root are encountered because of conjugation and inflection. In order to expand the original query to include the different form that query terms can be found, we tried three techniques.

First, we experimented with introducing the wildcard character into the original query terms. Results showed that using a naive approach of introducing the wildcard character earlier than the middle of the term resulted in a significant number of unrelated terms being introduced into the query. Moving the wildcard character toward the end of the term results in a significant increase of the relativeness of the added words. Applying an extra limitation on the length of the added words, resulted in an optimal performance, therefore showing that wildcard queries, can have substantial effect in IR tasks in the Greek language.

We also experimented with using word distrance and fuzzy similarity metrics. The initial results weren't as promising as it was hoped, mainly because a significant percent of expanded terms had different grammatical roots than the original query terms. We amended the phenomenon by requiring that the first half of the words is similar, in effect forcing that new terms have the same morphological root as the original terms. The results showed improvement, showing potential in the usage of such metrics. A more elaborate query expansion scheme may produce even better results but that was left for future work.

In conclusion, we demonstrated that although Greek is a very complex and intricate language, the usage of the above techniques can result in a significant increase of the quality of results in Web IR related tasks.

7. ACKNOWLEDGEMENTS

This paper is part of the 03ED404 research project, implemented within the framework of the "Reinforcement Programme of Human Research Manpower" (PENED) and co-financed by National and Community Funds (25% from the Greek Ministry of Development-General Secretariat of Research and Technology and 75% from E.U.-European Social Fund).

8. REFERENCES

[1] J. Allan, M. Connell, W. B. Croft, F. Feng, D. Fisher, and X. Li. Inquery and trec-9. In *Proceedings of TREC-9*, pages 551–577, 2000.

[2] K. Balog, L. Azzopardi, J. Kamps, and M. de Rijke. Overview of webclef. In *CLEF*, pages 803–819, 2006.

[3] M. W. Du and S. C. Chang. An approach to designing very fast approximate string matching algorithms. *IEEE Trans. on Knowl. and Data Eng.*, 6(4):620–633, 1994.

[4] B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Inf. Process. Manage.*, 36(2):207–227, 2000.

[5] F. Lazarinis. Engineering and utilizing a stopword list in greek web retrieval. *J. Am. Soc. Inf. Sci. Technol.*, 58(11):1645–1652, 2007.

[6] F. Lazarinis. Lemmatization and stopword elimination in greek web searching. In *EATIS '07*, pages 1–4, New York, NY, USA, 2007. ACM.

[7] V. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8–17, 1965.

[8] I. Macdonald C.; Lioma C.; Ounis. Terrier takes on the non-english web. In *iNEWS'07*, 2007.

[9] G. Ntais. Development of a stemmer for the greek language. Master's thesis, Stockholm University, 2006.

[10] P. Ogilvie and J. P. Callan. Experiments using the lemur toolkit. In *Text REtrieval Conference*, 2001.

[11] M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.

[12] B. Sigurbjornsson, J. Kamps, and M. de Rijke. Eurogov: Engineering a multilingual web corpus. In C. Peters, F. C. Gey, J. Gonzalo, H. Muller, G. J. F. Jones, M. Kluck, B. Magnini, and M. de Rijke, editors, *CLEF*, volume 4022 of *Lecture Notes in Computer Science*, pages 825–836. Springer, 2005.

[13] G. Tambouratzis. Automatic Corpora-based Stemming in Greek. *Lit Linguist Computing*, 16(4):445–466, 2001.

[14] L. Tzekou P.; Stamou S.; Zotos N.; Kozanidis. Querying the greek web in greeklish. In *iNEWS'07*, 2007.

Automatic Query Structuring from Sentences for Japanese Web Retrieval

Tetsuya Shibata
Department of Computer Science and Systems Engineering
Kobe University
1-1 Rokkoudai, Nada-ku, Kobe 657-8501, Japan
shibata@cs25.scitec.kobe-u.ac.jp

Koji Eguchi
Department of Computer Science and Systems Engineering
Kobe University
1-1 Rokkoudai, Nada-ku, Kobe 657-8501, Japan
eguchi@port.kobe-u.ac.jp

ABSTRACT

This paper proposes a query structuring method by analyzing Japanese natural language sentences that users input. We use linguistic structure information obtained by morphological analysis and dependency parsing on the natural language inputs. Our proposed method converts a sentence into a structured query on the basis of linguistic structures such as phrases and modification relations. Such structured queries enable effective retrieval with reasonable efficiency. To evaluate our proposed method, we compare it with the method using no structures at all, using a 100-gibabyte web collection mostly written in Japanese. We demonstrate through the experiments that mean average precision was improved about 8.6% using our query structuring method alone, and about 22.5% by combining our query structuring method with pseudo-relevance feedback.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation*

General Terms

Experimentation

Keywords

Web retrieval, term dependence, structured quries, Japanese information retrieval.

1. INTRODUCTION

A number of researches on query formulation for information retrieval have been conducted, to date. Automatic query formulation techniques can be categorized mainly into query expansion [6, 14, 1] and query structuring [9, 2]. The main focus of this paper is the latter. This paper also investigates the combination of query structuring and query expansion. While queries for information retrieval generally consist of a series of keywords, there are unignorable demands for retrieval with natural language queries. However, there are some severe problems of using Japanese natural language queries. Given a Japanese natural language sentence as a query, the query sentence needs to be decomposed into words because each Japanese word is not separated by a space delimiter. Even when a Japanese natural language input can be decomposed into words, directly using the set of the decomposed words as a query is not usually effective for retrieval. This is because not every word in a query sentence represents the user's information needs accurately. Moreover, the larger the number of words is, the more computational cost is usually required for retrieval. To solve these problems, we use linguistic structure information obtained by morphological analysis and dependency parsing on the natural language inputs. We structure a new query from a given query sentence using the detected dependencies and phrases. By doing this, we can generate structured queries reflecting dependencies between words and can remove verbose words. Such structured queries enable effective retrieval with reasonable efficiency. To evaluate our proposed method, we compare it with the method using no structures at all, using a 100-gibabyte web collection mostly written in Japanese. Moreover, we evaluate our query structuring method by combining with pseudo-relevance feedback, an automatic query expansion technique,

2. RELATED WORK

A series of retrieval methods using probabilistic language models have been proposed [7], such as query likelihood model [12, 4, 13] and relevance model [6]. Probabilistic language models give the probability of generating query words, usually on the basis of maximum-likelihood estimated distribution of words in each document. However, these methods need to be enhanced. These methods are commonly based on the "bag of words" assumption. This assumes that any word is independent of any other word. Therefore, most conventional methods disregard some important aspects, such as dependencies of words. However, analyzing a large document collection is not realistic because it is computationally expensive. To solve the problem described above, we use Markov random fields (MRFs) to model dependencies of words [9, 10]. The previous models assumed keyword queries, and so we need to extend the models to apply to natural language queries. This is because, when we directly use these previous models, the number of combinations of adjacent words used in computing MRF features explosively increases. Our proposed method is based on term dependence model using MRFs with linguistic structures in natural language queries, such as using phrases and modification relations. Besides our work, an extended term dependence model was investigated for Japanese information retrieval [2]. This paper is focused on Japanese natural language queries, while [2] was not the case. We make use of linguistic structures in Japanese natural language queries to formalize structured queries.

We briefly describe MRF-based retrieval model in Section 2.1. We then summarize dependency parsing that gives modification relations in the Japanese language in Section 2.2.

2.1 Term Dependence Model

According to [9], we briefly describe the term dependence model that uses Markov random fields (MRFs) for information retrieval.

MRFs are commonly used in the area of statistical machine learning to model joint distributions. Metzler and Croft proposed term dependence model that uses MRFs for query structuring and document ranking [9]. In this approach, MRFs are used to model the joint distribution $P_\Lambda(Q, D)$ over query terms $Q = \{q_1, ..., q_n\}$ and documents D, parameterized by Λ. MRFs are constructed from a graph G, where the nodes represent the random variables and the edges represent the existence of dependency. MRFs assume the Markov property under which each node is independent from all of its non-adjacent nodes. The MRF-based retrieval model assumes that G consists of query nodes q_i and a document node D. The joint distribution $P_\Lambda(Q, D)$ can be defined by:

$$P_\Lambda(Q, D) = \frac{1}{Z_\Lambda} \prod_{c \in C(G)} \psi(c; \Lambda) \qquad (1)$$

where $Q = \{q_1, ..., q_n\}$, $C(G)$ is the set of cliques in G, each ψ is a non-negative potential function over cliques parameterized by Λ, and $Z_\Lambda = \sum_{Q,D} \prod_{c \in C(G)} \psi(c; \Lambda)$ normalizes the distribution. The joint distribution is uniquely defined by the graph G, the potential functions ψ, and the parameter Λ.

To rank the documents, the probability of the documents can be computed as obtained above, given a query. Ranking according to the probability is equivalent to that of its logarithm. We use "$\overset{\text{rank}}{=}$" to express such equivalence. When ranking documents, $P_\Lambda(Q)$ that is probability of query Q can be thought of as constant. Therefore, $P_\Lambda(D|Q)$ can be modified by:

$$\begin{aligned} P_\Lambda(D|Q) &= \frac{P_\Lambda(Q, D)}{P_\Lambda(Q)} \\ &\overset{\text{rank}}{=} \log P_\Lambda(Q, D) - \log P_\Lambda(Q) \\ &\overset{\text{rank}}{=} \sum_{c \in C(G)} \log \psi(c; \Lambda) \qquad (2) \end{aligned}$$

All potential functions are non-negative and commonly parameterized as:

$$\psi(c; \Lambda) = \exp\Big(\lambda_c f(c)\Big) \qquad (3)$$

where $f(c)$ is a real-valued feature function over clique, and λ_c is the weight on a particular feature function. Using this equation, finally, the following ranking function is obtained.

$$P_\Lambda(D|Q) \overset{\text{rank}}{=} \sum_{c \in C(G)} \lambda_c f(c) \qquad (4)$$

The following are examples of three assumptions of term dependence.

- Full independence:
  the assumption that all query terms are independent of each other.
- Sequential dependence:
  the assumption that neighboring query terms are dependent on each other.
- Full dependence:
  the assumption that all query terms are dependent on each other.

To express these assumptions, the following ranking function can be derived:

$$P_\Lambda(D|Q) \overset{\text{rank}}{=} \sum_{c \in T} \lambda_T f_T(c) + \sum_{c \in O} \lambda_O f_O(c) + \sum_{c \in O \cup U} \lambda_U f_U(c) \qquad (5)$$

where T is a set of cliques that consist of a document node and a query node. O is a set of cliques that consist of a document node and two or more query nodes that appear in sequential order in the query. U is a set of cliques that consist of a document node and two or more query nodes that appear non-contiguously within the query. The graphical model that represents these assumptions is shown in Figure 1.

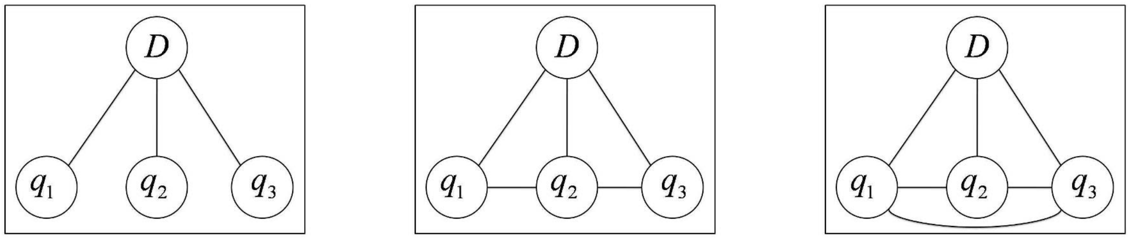
2.2 Japanese Dependency Parsing

We assume a Japanese natural language sentence as a query. A query sentence is needed to be decomposed into terms to formulate an information retrieval query. We use Japanese morphological analysis to parse a query sentence into a series of morphemes (roughly speaking, building units of language expression that convey a meaning or a linguistic function) and to tag part-of-speech classes. We consider each detected morpheme as a word. In this paper, we used "ChaSen"[1] for the morphological analysis.

In Japanese, a sentence can be decomposed into morphemes or phrases (i.e., "bunsetsu" in Japanese), each of which consists of several morphemes. Each phrase usually modifies another phrase in the sentence. This kind of linguistic structure is called "modification relation", "modification structure" or "kakariuke-dependency". We conduct dependency parsing to detect the modification structure. When modification structure is detected, dependency between phrases is found. We structure queries with the modification relation between phrases. An example of modification structure is shown in Figure 2. In this example, "→" indicates a modification relation and "=" indicates apposition-and-parallel relation which is a particular case of modification relation. Moreover, "/" indicates a word delimiter in each phrase, the reading of a phrase is shown in "[·]", and the English translation of a phrase is shown in "(·)". In Japanese, each phrase usually modifies another phrase followed (not necessarily successive), as you can see in this example. We treat apposition-and-parallel relations differently

[1]http://chasen.naist.jp/hiki/ChaSen/

Figure 1: Examples of Markov random field models for three query terms under three assumptions: (left) full independence (middle) sequential dependence (right) full dependence.



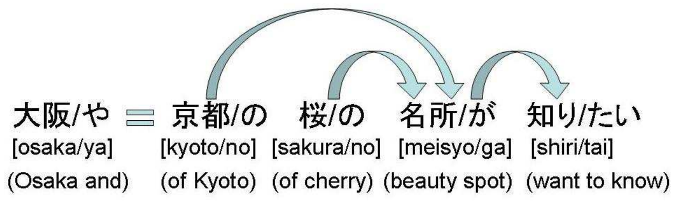


Figure 2: An example of modification structure of a Japanese sentence consisting of five phrases. In this example, "/" indicates the word-separator in a phrase, the reading of a phrase is shown in "[·]", and the English meaning of a phrase is shown in "(·)".

from modification relations in our proposed methods that we will describe later. In this paper, we used "CaboCha"[2] [5] as a dependency parsing tool.

3. AUTOMATIC GENERATION OF STRUCTURED QUERIES

In this section, we introduce our proposed method that was an extension of Metzler's term dependence model described in Section 2.1. Metzler's term dependence model assumed a keyword query that consists of a small number of words. Therefore, this model does not work efficiently with a natural language query that usually consists of a larger number of words. We generate a structured query using the results of applying linguistic analysis to a query sentence. In our proposed method, we make use of three kinds of linguistic information: modification relations, phrases and word classes.

As previously mentioned in Section 2, the proposed method is based on the framework of the MRFs. Our method assumes the graph that consists of query nodes and a document node, similarly to Metzler's term dependence model. Here, the query nodes are corresponding to morphemes decomposed from a natural language query. For simplicity, we use the expressions "term" or "word" instead of "morpheme" hereafter. Due to an increasing number of query nodes extracted from natural language queries, computational cost explosively increases if we compute feature quantities with any cliques of adjacent nodes, as required in Metzler's term dependence model described in Section 2.1. To address such problems, we make use of modification relations and word classes to reduce the number of cliques with which feature quantities are computed. We use only cliques that consist of linguistically specified modification relations between query nodes. We assumed that there are no relations between the other combinations of query nodes. We removed words that are assigned some word class expected not to be effective for retrieval. These constraints enable us to reduce computational cost. We also take into account that linguistically detected phrases as phrase proximity queries, because words in linguistic phrases usually have tight dependencies each other.

Based on these ideas, we now describe three dependence assumptions, as follows.

- Full independence (FI):
  the assumption that all query terms are independent of each other.
- Phrase dependence (PD):
  the assumption that neighboring query terms in a phrase are dependent on each other.
- Modification dependence (MD):
  the assumption that query terms in a modification relation are dependent on each other.

To express these assumptions, we derive the following ranking function:

$$P_\Lambda(D|Q) \overset{\text{rank}}{=} \sum_{c\in T}\lambda_T f_T(c) + \sum_{c\in P}\lambda_P f_P(c) + \sum_{c\in M}\lambda_M f_M(c) \quad (6)$$

where T is a set of cliques that consist of a document node and a query node having the following word classes: noun, number, English or unknown class. P is a set of cliques that consist of a document node and two or more query nodes that are successive nouns within each linguistic phrase. M is a set of cliques that consist of a document node and two or more query nodes that are combinations of a term or successive terms in a modifying phrase and in a modified phrase. We regard suffix/prefix and the corresponding modified word as one phrase because dependency between them usually tighter. We did not use the cliques when a phrase consists of only words other than nouns, verbs, adjectives, English, numbers or unknown classes. In the case of an apposition-and-parallel relation, we use the clique assuming all apposition/parallel phrases involved in a modification

[2]http://chasen.org/ taku/software/cabocha/

relation when an apposition/parallel phrase appears in the modification relation. In our method, we constrain cliques with which feature quantities are computed, as described above. We enforce the constraint $\lambda_T + \lambda_P + \lambda_M = 1$. The graphical model that represents these assumptions is shown in Figure 3. In this example, there are a couple of modification relations between phrase1 and phrase3, and between phrase2 and phrase3.

4. QUERY EXPANSION WITH RELEVANCE MODEL

Relevance model was proposed in [6], and reported that using relevance model as pseudo relevance feedback was effective for Web retrieval [11, 1].

We first describe the relevance model. Assuming that a set of documents are split into relevant class and non-relevant class, relevance model can be estimated from set of relevant documents. Once relevance model is estimated in response to a query, we can retrieve other relevant documents using relevance model to formulate a new query. However, it is usually not easy to completely figure out the relevant class. Therefore, the relevance model is usually estimated, in a nonparametric manner, only with a query given by a user and a target document collection. Given a query $Q = \{q_1, ..., q_n\}$, probability of word w observed from the relevance model R is estimated as follows:

$$P(w|R) = \sum_{D \in C} P(w|D)P(D|q_1, ..., q_n) \tag{7}$$

Practically, the relevance model $P(w|R)$ is approximated as follows. A mixed distribution is constructed by Equation (7) using top N documents ranked by $P(D|q_1, ..., q_n)$. $P(w|R)$ is then approximated using M words with highest probability from the mixed distribution obtained above.

In this paper, we further use a combination of the structured queries that we proposed and the queries obtained by the relevance model [11]. A set of documents are first retrieved using our structured queries using Equation (6). We then estimate the relevance model from top-ranked documents using Equation (7) in the approximated manner mentioned above. We combine the features noted in Equation (6) (which will be denoted as Q_{st} in Section 5) and the additional features obtained by the relevance model: $\sum_{c \in RM} \lambda_{RM} f_{RM}(c)$ (which will be denoted as Q_{rm} in Section 5), as follows.

$$P_\Lambda(D|Q) \overset{\text{rank}}{=} \sum_{c \in T} \lambda_T f_T(c) + \sum_{c \in P} \lambda_P f_P(c) + \sum_{c \in M} \lambda_M f_M(c) + \sum_{c \in RM} \lambda_{RM} f_{RM}(c) \tag{8}$$

where RM is a set of cliques that consist of a document node and a node of an individual term that appears in the approximated relevance model, and λ_{RM} for an individual feedback term is determined to be proportional to the approximated $P(w|R)$.

The graphical model that represents the three assumptions combined with the additional features obtained by the relevance model is shown in Figure 4. In this example, $fb_1 \cdots fb_n$ indicate the additional "feedback" features obtained by the relevance model.

5. EXPERIMENTS

5.1 Experimental Setting

We carry out experiments to evaluate the proposed methods. In the proposed methods, we structure queries using linguistically analyzed result of natural language queries. We compare the proposed model with Full Independence (FI) model that assumes query terms are independent on each other.

Before we detail the experiments, we briefly describe the retrieval platform "Indri" [8]. In this paper, we use Indri as a retrieval platform, such as for indexing with proximity search; however, our proposed methods can be applied on any search engines that provide proximity operators.

We used the following two query expressions, according to Indri query language.

- #1($\cdot$) : the terms appear sequentially in the specified order.
- #uwN($\cdot$) : the terms appear in any order within a window size N.

In our proposed methods, we use the query expression #1($\cdot$) in PD model, and #uwN($\cdot$) in MD model. We set the window size N for #uwN($\cdot$) by (the number of query terms) $\times$ (per-word window size). We empirically determined the parameter of the per-word window size. We performed pseudo-relevance feedback using the relevance model described in Section 4 in the following manner in Indri query expression.

$$Q_{new} = \#\text{weight}(\nu Q_{st} \quad (1.0 - \nu) Q_{rm}) \tag{9}$$

where ν is a parameter, and #weight is Indri query operator to weight query terms. Q_{st} indicates the proposed structured queries and Q_{rm} the feedback queries obtained by pseudo-relevance feedback. We performed retrieval with the expanded query obtained by Equation (9). Examples of structured queries we use in experiments are shown in Table 1. In this example, we treat a phrase as one term.

For experiments, we used NTCIR-3 WEB [3] collection that was designed for evaluating Japanese Web retrieval. The NTCIR-3 WEB collection consists of 100-gigabyte data (which are mainly gathered from .jp domain and mostly written in Japanese or English), the topics that are statements of information needs (written in Japanese) and a list of relevance judgment results with regard to each topic. In this paper, we empirically determined parameters over the test collection above, and evaluate the proposed methods, for preliminary investigations. We use description field of the topics (specified with "DESC" tag) as a natural language queries. The number of the topics is 47. We formulated structured queries for each topic using our proposed methods. We also evaluated the combination of our proposed structured queries and the pseudo-relevance feedback, as described in Section 4. We used Mean Average Precision —non-interpolated— (MAP) as an evaluation metric.

We used two kinds of stop words or expressions for generating queries, as follows.

- Phrases that are specific in natural language queries and are often not to be effective for retrieval, such as "shiritai" (want to know) or "sagashitai" (want to search for).

Figure 3: Examples of Markov random field models for four terms decomposed from a query under three assumptions: (left) full independence (middle) phrase dependence (right) modification dependence. In these examples, there are a couple of modification relations between phrase1 and phrase3, and between phrase2 and phrase3.

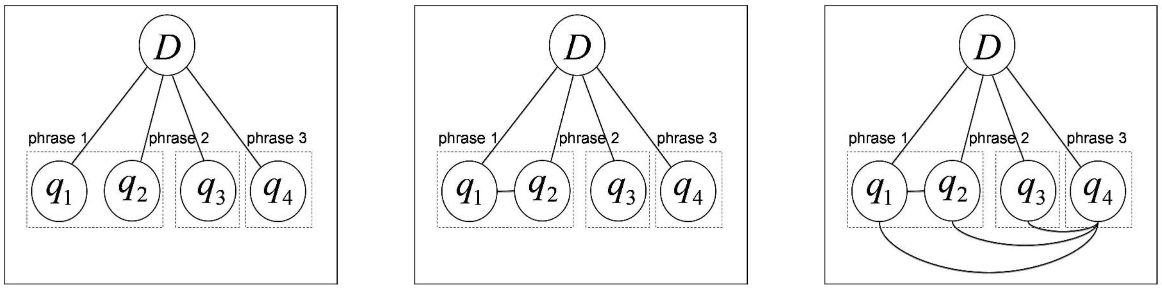


Figure 4: Examples of Markov random field models for four terms decomposed from a query with n feedback terms under three assumptions: (left) full independence (middle) phrase dependence (right) modification dependence.

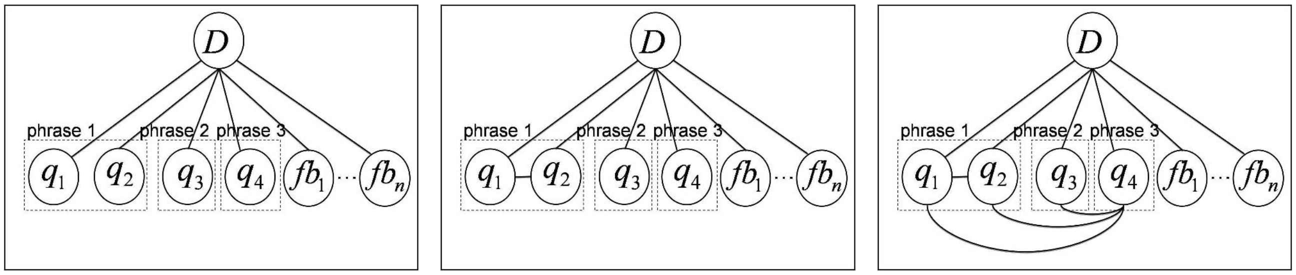


Table 2: Experimental results of FI+PD+MD model in terms of MAP, varying per-word window size of term dependence.

window size	MAP
2	0.1643
5	0.1655
10	0.1680
20	0.1664
50	0.1662

- Single one-byte alphabetical/numerical characters, single one- or two-byte symbol characters, single Japanese "hiragana" characters, single Japanese "katakana" characters, and all possible pairs of single "hiragana" characters.

5.2 Experimental Results

In the first experiment, we investigated appropriate size of the unordered window used in our MD model. The evaluation results of the total model of FI+PD+MD in terms of MAP are shown in Table 2, varying the unordered window size in the MD model. According to Table 2, MAP increased when per-word window size is up to 10 and decreased when the window size gets wider. Therefore, we used the per-word window size to 10 in the following experiments.

As described in Section 3, we proposed the phrase dependence (PD) model assuming that adjacent terms appeared in a linguistic phrase in a natural language query are dependent on each other, and the modification dependence (MD) model assuming that terms appeared in a modification relation are dependent on each other. In the experiments, we used combination of the models: FI+PD, FI+MD and FI+PD+MD. Here, the full independence (FI) model is assumed that all query terms are independent on each other. We experimented to get appropriate values of weighting parameters in each model. According to the results, the best parameters of FI+PD model were $(\lambda_T, \lambda_P) = (0.95, 0.05)$, the best parameters of FI+MD model were $(\lambda_T, \lambda_M) = (0.9, 0.1)$, and the best parameters of FI+PD+MD model were $(\lambda_T, \lambda_P, \lambda_M) = (0.9, 0.05, 0.05)$. In the experiments above, we determined the optimal parameters by varying each parameter in 0.1 step and then varying these in 0.05 step in a specified range. With these parameters, the experimental results comparing the proposed models with the baseline FI model are shown in Table 3. According to Table 3, comparing with FI model, FI+PD improved up to 4.40%, FI+MD improved up to 7.95%, and FI+PD+MD improved up to 8.60%. Comparing with FI model that assumed all query terms are independent of each other, our proposed models using linguistic phrases and modification relations significantly improved retrieval effectiveness. These results suggested that information on linguistic phrases and modification relations are effective for retrieval with a natural language query.

Table 1: Examples of structured queries. For each Japanese query that we used, an English translation is followed. In the natural language query, "/" indicates the word-separator in each phrase, and "_" indicates the phrase-separator in each sentence.

natural language query	nihon/no_jidousha/no_shourai/zou/ha_dou_nat/te/iru/no/ka_siri/tai
(English translation)	I want to know future vision of Japanese car
FI model	#combine(nihon jidousha shourai zou)
(English translation)	#combine(Japanese car future vision)
FI+PD+MD model	#weight(0.9 #combine(nihon jidousha shourai zou) 0.05 #combine(#1(shourai zou)) 0.05 #combine(#uw20(nihon jidousha) #uw20(jidousha #1(shourai zou))))
(English translation)	#weight(0.9 #combine(Japanese car future vision) 0.05 #combine(#1(future vision)) 0.05 #combine(#uw20(Japanese car) #uw20(car #1(future vision))))

Table 3: Experimental results with MAP using various models.

Model	MAP	%increase
FI	0.1547	0.00
FI + PD	0.1615	4.40
FI + MD	0.1670	7.95
FI + PD + MD	0.1680	8.60

Additionally, we experimented with combination of our proposed structured queries above and pseudo-relevance feedback in Section 4. We changed the number of documents N and the number of feedback terms M in pseudo-relevance feedback with 5, 10 and 20, to determine appropriate parameters. We also changed the weighting parameter of ν that appears in Equation (9). According to the results, the best number of documents N, the best number of queries M, the best weighting parameter ν in pseudo-relevance feedback were $(N, M, \nu) = (5, 20, 0.7)$ for FI model, $(5, 20, 0.5)$ for FI+PD model, $(10, 5, 0.7)$ for FI+MD model, and $(10, 10, 0.5)$ for FI+PD+MD model. With these parameters, experimental results of pseudo-relevance feedback with our proposed structuring model and with baseline FI model are shown in Table 4. Experimental result with each of these combinations of parameters is shown in Table 5. Here, %increase$_{fd}$ indicates a percentage improvement achieved by pseudo-relevance feedback, and %increase$_{total}$ indicates a total percentage improvement comparing with baseline FI model without feedback.

According to Table 4, in pseudo-relevance feedback environment, our structured queries improved up to 15.06% in terms of MAP, comparing with FI model on the same condition. Our proposed structuring models with feedback improved up to 22.50%, comparing with FI model without feedback. By using pseudo-relevance feedback, the proposed models improved up to 12.80%, while the baseline FI model improved only 6.46%. Therefore, our proposed structuring models achieved further improvements in the environment of pseudo-relevance feedback. At this point, we confirmed that our proposed structuring methods can achieve more sophisticated retrieval than the others, since pseudo-relevance feedback is sensitive to the initial retrieval results.

Figure 5 indicates the results of topic-by-topic evaluation

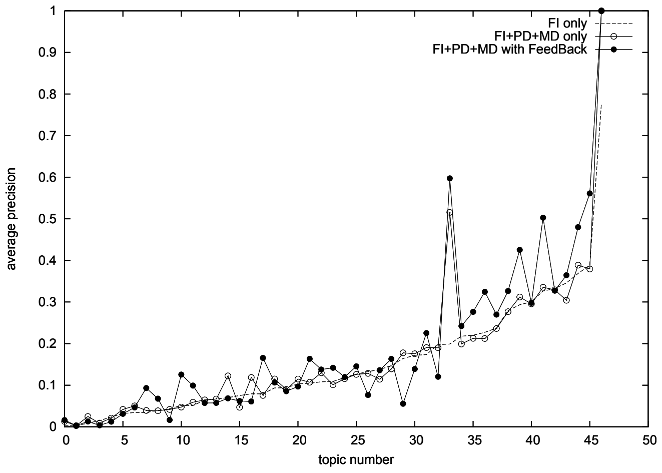


Figure 5: Topic-by-topic evaluation.

for FI+PD+MD model with/without feedback, compared with FI model without feedback. In the horizontal axis, the topics are sorted in order of average precision values of FI model without feedback. As you can see in this graph, our proposed model without feedback improved retrieval effectiveness for some topics and was comparable with the baseline FI model for most of the other topics. In pseudo-relevance feedback environment, our proposed model further improved retrieval effectiveness for most of the topics, while for some topics pseudo-relevance feedback hurt retrieval effectiveness.

6. CONCLUSIONS

In this paper, we newly proposed several methods to structure queries using linguistic information of natural language queries, such as modification relations, for Japanese information retrieval. We indicated problems that natural language queries contain words that are not related to content of users' information needs, and that previous term dependence models involve unaffordable computational cost for such long queries. We automatically generated structured queries, reflecting linguistic dependencies between words and removing verbose words. Such structured queries achieve ef-

Table 4: Experimental results with best parameters of pseudo-relevance feedback. "*" indicates statistically significant improvements over the FI model alone, where $p < 0.05$ using the two-sided Wilcoxon signed-rank test.

Model	MAP	MAP(after feedback)	%increase$_{fd}$	%increase$_{total}$
FI	0.1547	0.1647*	6.46	6.46
FI + PD	0.1615	0.1781*	10.28	15.13
FI + MD	0.1672	0.1831*	9.51	18.36
FI + PD + MD	0.1680	0.1895*	12.80	22.50

Table 5: Detailed experimental results with various parameters of pseudo relevance feedback. "%increase$_{fd}$" indicates percentage improvement achieved by pseudo-relevance feedback, and "%increase$_{total}$" indicates a total percentage improvement comparing with baseline FI model without feedback. "*" indicates statistically significant improvements over the FI model alone, where $p < 0.05$ using the two-sided Wilcoxon signed-rank test.

Model	(N, M, ν)	MAP	%increase$_{fd}$	%increase$_{total}$
FI	FI only	0.1547	0.00	0.00
	(5,5,0.7)	0.1586	2.52	2.52
	(5,10,0.7)	0.1626*	5.11	5.11
	(5,20,0.7)	0.1647*	6.46	6.46
	(10,5,0.9)	0.1568	1.36	1.36
	(10,10,0.9)	0.1581*	2.20	2.20
	(10,20,0.9)	0.1562*	0.97	0.97
	(20,5,0.9)	0.1601	3.49	3.49
	(20,10,0.9)	0.1627*	5.17	5.17
	(20,20,0.9)	0.1614*	4.33	4.33
FI + PD + MD	FI + PD + MD only	0.1680	0.00	8.60
	(5,5,0.7)	0.1749	4.11	13.06
	(5,10,0.7)	0.1754*	4.40	13.38
	(5,20,0.7)	0.1767*	5.18	14.22
	(10,5,0.7)	0.1841*	9.58	19.00
	(10,10,0.5)	0.1895*	12.80	22.50
	(10,20,0.5)	0.1825*	8.63	17.97
	(20,5,0.9)	0.1785*	6.25	15.38
	(20,10,0.9)	0.1789*	6.49	15.64
	(20,20,0.9)	0.1778*	5.83	14.93

fective retrieval with reasonable efficiency. Comparing with fully unstructured queries, our structured queries improved up to 8.6% in terms of mean average precision. We also evaluated combination of our structured queries with pseudo-relevance feedback. We demonstrated that our structured queries improved up to 12.80% when using pseudo-relevance feedback. In the pseudo-relevance feedback environment, our structured queries improved up to 15.06%, comparing with unstructured queries on the same condition. Furthermore, our proposed models with feedback improved up to 22.50% compared with unstructured queries without feedback. These evidences through preliminary investigations suggest that our structured queries using linguistic information achieve significant improvement from unstructured queries in retrieval effectiveness. Finally, we also obtained an insight that the linguistic information such as modification relations and phrases are useful for retrieval with a natural language query.

Acknowledgements

This work was supported in part by the Grant-in-Aid for Scientific Research on Priority Areas "Info-plosion" (#19024055), Exploratory Research (#18650057), and Scientific Research (B) (#20300038) from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

7. REFERENCES

[1] F. Diaz and D. Metzler. Improving the estimation of relevance models using large external corpora. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, pages 154–161, Seattle, Washington, USA, 2006.

[2] K. Eguchi and W. B. Croft. Query structuring with two-stage term dependence in the Japanese language. In *Information Retrieval Technology: Third Asia Information Retrieval Symposium*, volume 4182 of *Lecture Notes in Computer Science*, pages 522–529. Springer, 2006.

[3] K. Eguchi, K. Oyama, E. Ishida, N. Kando, and

K. Kuriyama. Overview of the Web Retrieval Task at the Third NTCIR Workshop. In *Proceedings of the 3rd NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering*, Tokyo, Japan, 2003.

[4] D. Hiemstra. A linguistically motivated probabilistic model of information retrieval. In *Research and Advanced Technology for Digital Libraries*, volume 1513 of *Lecture Notes in Computer Science*, pages 569–584. Springer-Verlag, 1998.

[5] T. Kudo and Y. Matsumoto. Japanese dependency analysis using cascaded chunking. In *Proceedings of the 6th Conference on Natural Language Learning*, pages 63–69, Taipei, Taiwan, 2002.

[6] V. Lavrenko and W. B. Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, pages 120–127, New Orleans, Louisiana, USA, 2001.

[7] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[8] D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Information Processing and Management*, 40(5):735–750, 2004.

[9] D. Metzler and W. B. Croft. A Markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, pages 472–479, Salvador, Brazil, 2005.

[10] D. Metzler and W. B. Croft. Latent concept expansion using Markov random fields. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, pages 311–318, Amsterdam, The Netherlands, 2007.

[11] D. Metzler, T. Strohman, H. Turtle, and W. B. Croft. Indri at TREC 2004: Terabyte Track. In *Proceedings of the 13th Text Retrieval Conference (TREC 2004)*. NIST Special Publication 500-261, 2004.

[12] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, Melbourne, Australia, 1998.

[13] F. Song and W. B. Croft. A general language model for information retrieval. In *Proceedings of the 8th ACM International Conference on Information and Knowledge Management*, pages 316–321, Kansas City, Missouri, USA, 1999.

[14] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the 10th ACM International Conference on Information and Knowledge Management*, pages 403–410, Atlanta, Georgia, USA, 2001.

Query Selection for Improved Greek Web Searches

Sofia Stamou Lefteris Kozanidis Paraskevi Tzekou Nikos Zotos
Computer Engineering Department, Patras University 26500 Greece
{stamou, kozanid, tzekou, ztoson} @ceid.upatras.gr

ABSTRACT
As the Web becomes an integral part of our everyday life and the Internet-literate population grows rapidly, the Search Engine market is steadily gaining a high monetary value. Unfortunately, today, the distribution of the search market share is dominated by English-speaking users and stakeholders, basically because English is the lingua franca of the Web. Thus, although the majority of the Web users are non-English native speakers, they naturally gravitate to using English in order to explore the plentiful Web content. In this paper, we propose a query selection mechanism for assisting users perform successful non-English Web searches. Our mechanism combines linguistic analysis and Web mining techniques and aims at assisting users select informative and well-specified queries for expressing their information needs in languages other than English. Our technique is validated on a dataset of 70 Greek queries issued to Google search engine over a period of 3 weeks. Obtained results demonstrate that our query selection mechanism yields improved retrieval performance compared to existing non-English search strategies and as such we believe that it can be fruitfully deployed for other natural languages.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Query formulation, Search process, Retrieval models, Selection process; H.3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness).

General Terms
Performance, Design, Experimentation.

Keywords
Greek web search, query selection, linguistic analysis, text mining, semantics.

1. INTRODUCTION
The most popular way for finding information on the Web is go to a search engine, issue a keyword query that describes an information need and receive a list of results that somehow relate to the information sought. Despite the recent advances in Web search there are two main challenges that a search engine needs to deal with in order to satisfy all user search requests. The first challenge is to effectively process the multilingual Web content in order to be able to serve multilingual queries and the second is to be able to suggest users with alternative queries in case their issued keywords fail to retrieve the desired information.

For the first challenge, a search engine needs not only accurately detect the query language and look for information in a multilingual index, but it also needs to account for the language properties and usage. To tackle such problems there have been previous studies that investigate Web searching through non-English queries (cf. to [3] for an overview). The striking majority of these studies concentrate on either searching the Web in a particular language (other than English) [26] [5] [14] [15] [17], or on cross-lingual Web information retrieval [20] [10] [6]. The most popular approach towards cross-lingual Web information retrieval is to explore bilingual corpora or dictionaries in order to translate the query and/or the documents from one language to the other, overcoming thus language barriers. Despite the usefulness of Machine Translation techniques in a multilingual Web search setting, these are intrinsically insufficient in a practical deployment because of their implementation and maintenance high cost, their limited availability and their complete dependence on the resources' accuracy and lexical completeness. With respect to the second challenge, i.e. how to help users specify informative queries, researchers have studied the improvement of the user issued queries with semantically similar terms [4] [9]. Query refinement is the process of providing users with alternative query formulations in the hope of retrieving relevant information. Besides query refinement, researchers have also studied ways of personalizing search results according to the user interests [32]

One aspect that none of the reported studies addresses is how to assist non-English Web searchers select informative queries that are both expressive of their search interests and semantically related to their typed keywords. In this paper, we investigate the problem of assisting Greek Web users select alternative query wordings for expressing their search intentions. In particular, we present a personalized query selection mechanism that employs linguistic analysis and Web mining techniques in order to identify good query alternatives in Greek Web searches.

Our model relies on the intuition that a user's search behavior and querying patterns are common throughout her different Web searches, irrespectively of the language employed. Therefore, we suggest the uniform mining of the user's previously collected search trace in order to derive the user interests and the user preferred queries for expressing those interests. Based on the identified user interests and interest-expressive queries, we employ linguistic analysis in order to firstly decipher the intention of the user's current query and thereafter improve it with alternative keywords that match both the query semantics and the user interests. We applied our query selection mechanism to a number of experimental Greek language queries that have been issued to

Google search engine and we evaluated the accuracy of the search results. Obtained results demonstrate that our query selection mechanism yields improved retrieval performance compared to existing search strategies and as such we believe that it can be fruitfully deployed for other natural languages.

The rest of the paper is organized as follows. We begin our discussion with a brief introduction to the particularities associated with Greek Web searches. In section 3, we present our proposed query selection mechanism and we describe the way in which we mine the user's search logs in order to derive the user interests and preferred search keywords (section 3.1). We then discuss how we explore the identified interests for deciphering the user's current query and based on this knowledge we show how our model picks alternative query wordings for refining search (section 3.2). In section 4, we present our experimental evaluation and we discuss obtained results. We conclude the paper in section 5.

2. SEARCHING THE GREEK WEB

This section provides a brief overview on the Greek Web, the characteristics of the Greek language and how search engines handle Greek language queries.

2.1 The Greek Web

The prime difficulty associated with searching the Greek Web is to accurately define the latter, i.e. to detect the *boundaries* of its graph so as to be able to download its content and subsequently head Greek language searches against it. The straightforward claim that the Greek Web is composed of sites registered under the .gr domain is misleading, since many Greek sites are hosted under the .net, .com, or .org top-level domains [16]. In addition, many sites in the .gr domain contain English content that is Greek-oriented. Under stricter criteria, the Greek Web is defined as the collection of Web pages written in Greek. Despite some recent attempts to capture the entire picture of the Greek Web, the latter remains incomplete as we would need to equip web crawlers with automatic language detection tools in order to focus their Web visits on Greek content alone. Nevertheless, when it comes to the .gr domain, the extend of our knowledge is much better thanks to the study of [1] who showed that a breadth-first crawling for up to 5 levels for dynamic and up to 15 levels for static pages resulted into 4 million pages as of January 2004. Moreover, [13] found that as of mid 2007 there were 73,400 distinct sites in the .gr domain and that the Greek Web's weighted graph, when accounting only for external (i.e. across domain names) links reached up to 76,506 nodes and 141,791 edges. For a comprehensive overview on the characteristics of the .gr domain, we refer the interested reader to the work of [2].

2.2 The Greek Language

Greek is a highly inflected Indo-European language that uses an alphabet of 24 upper case and 25 lower case letters and a number of accents, depending on the usage of a given wordform. Greek has three main forms, namely Classical Greek (also known as Ancient Greek), Katharevousa (an imitation of Classical Greek that was mainly used in formal writings during the 19th and the 20th centuries) and Dimotiki (what is known as Standard Modern Greek), which is currently (from 1976 onwards) used by Greek speakers in all their communications, verbal and written. Another characteristic of the Greek language is its rich inflectional morphology, which is attested in the numerous wordforms[1] that might correspond to a single lemma. When it comes to orthography, Greek writing is much complicated not only due to the different ways of spelling the same sounds, but also due to the wide use of diphthongs and digraphs, including various pairs of vowel letters.

Another issue that adds to the complexity of understanding and using Greek concerns the transliteration of Greek to Latin letters; a practice so common in the digital era that has given rise to the formation of a hybrid language, known as Greeklish[2]. Greeklish emerged as a convenient mean for e-writing Greek in operating systems and applications with no support of the Greek character set. Today, modern software supports Greek but still it is much easier for Greek computer literates to e-write in Greeklish because it is faster to type and they do not have to worry for orthography.

Based on the above, it becomes evident that the computational processing of Greek is a complex task that requires extensive linguistic knowledge. When it comes to the Web search paradigm, handling and understanding Greek is a challenging task that has attracted the interest of many researchers, as we present next.

2.3 How Search Engines Respond to Greek Language Queries?

Stimulated by the characteristics of the Greek language and the Web's evolution in non-English domains, many researchers [17] [7] [28] [18] have studied the problems associated with searching the Web via Greek queries and they have come up with some interesting observations. Lazarinis [17] studied how different search engines respond to Greek queries and found that there is great variation in the handling of Greek between global and local search engines. In particular, he found that global search engines, apart from Google, are case sensitive, they do not apply stopword removal and stemming for Greek and as such they hinder the retrieval of pages that contain the query terms in a slightly different form compared to the user-typed keywords. Although global search engines ignore the characteristics of the Greek language, [7] experimentally showed that they outperform Greek search engines. Their study on a set of 309 navigational queries issued against five global and five Greek search engines revealed that the global engines have higher success rates, ranging from 48.54% to 73.79% than the local engines, which range from 10.68% to 52.43%. Nevertheless, the rate of success implies that there is much room for improvement towards both the Greek Web's coverage and the Greek queries' handling.

Besides the comparison between global and local search engines, researchers have studied ways of increasing the engines' capability in handling Greek queries. In [18] the authors investigated the incorporation of a morphological normalizer into the query processor of the Greek search engine Anazitisis[3] and found that morphologically normalized queries although they slightly drop recall, they significantly increase precision of the search results. In [27] the authors implemented a query expansion module that relies on the Greek WordNet [8] in order to firstly disambiguate queries and thereafter enrich them with synonymous terms. They validated their module on a small set of 18 queries and found that

[1] In Modern Greek a noun might have up to 7 distinct wordforms, whereas a verb might have up to 150 different wordforms.

[2] Greeklish is a blend of the words Greek and English.

[3] http://www.anazitisis.gr

the success of expansion depends upon the type of the query and it is more pronounced for long (more than 2 terms) queries.

In a different approach [29] studied the expansion of Greek Web queries with terms that are extracted from the query retrieved pages that the user regularly revisits across her different searches. A study on a small set of 7 search sessions showed that improved queries achieve higher retrieval precision compared to non-expanded queries. Our study, although complementary to this work, is different in that: (i) we select alternative query terms based not only on the semantics of the user interesting pages but also on the user preferred keywords for verbalizing those interests and (ii) we rely on the user's multilingual past searches for deriving a complete user search profile.

Recently, a human study on Greeklish Web searches [28] showed that 40.5% of the 42 study participants use Greeklish queries when their Greek searches fail to retrieve the desired information and that 67.5% of the participants would like to receive both Greek and Greeklish data in the results of their Greeklish queries. Based on the above observations, the authors implemented a converter and run an experiment in which they expanded Greek queries with their Greeklish equivalents and vice versa and found that blending Greeklish and Greek yields increased retrieval relevance compared to Greek-only and Greeklish-only Web searches.

Summarizing, reported works on querying the Greek Web lead to the conclusion that there is still a lot of work to be done before the Greek-speaking search engines become as powerful as their English counterparts. In light of the above, we have implemented a query selection module that aims at assisting Greek Web searchers specify *good* queries and which we present next.

3. QUERY SELECTION FOR IMPROVED GREEK WEB SEARCHES

The design and implementation of our query selection mechanism relies on the assumption that a user employs uniform query patterns throughout her Web searches irrespectively of the search engine used or the language preferred for verbalizing her information needs. With that in mind and considering that a large number of Greek searchers use global search engines to which they submit both Greek and English queries, we have designed a model that mines the user's previous searches (both Greek and English) in a uniform manner in order to derive both the user interests and the queries preferred for expressing those interests. The basic steps, upon which our model operates (depicted in Figure 1), are:

1) Collect the user's query logs (English and Greek) and process them to identify the query concepts. Rely on the identified concepts to derive the user topical interests and deduce the queries preferred by the user for expressing those interests.
2) Translate to Greek the concepts that represent the user preferred topics and the topic-preferred queries and store them locally in a utility index.
3) Given a user's current (Greek)[4] query, process it, lemmatize it and map it against its corresponding WordNet nodes.
4) Compute semantic similarity between the query matching senses and the senses used to describe the user's interests and preferred queries in the utility index.
5) Pick the sense of the maximum similarity as the query sense and use its synonyms and previously preferred keywords for improving the current query.

Before we proceed with the detailed description of each individual component and the way in which these have been integrated into a common infrastructure, let's first examine the validity of our underlying assumption, i.e. that user search habits are consistent across multilingual searches.

A number of studies [24] [25] have shown that although there is great variation in the way people from different regions search the Web, nevertheless this variety in the search trends is due to cultural and social issues and has little to do with the language employed for online searches. While searchers vary in their habits and topical preferences, there is great uniformity in the way people query the Web. Based on the above, we may conclude that the search habits of a user with specific information interests remain relatively stable across her different Web searches. Moreover, a large-scale tracking survey [19] on people's use of the internet revealed that the majority of Web users stick to just one search engine for their online searchers and that 99% of the users prefer to launch a new search on the same (preferred) engine than switch to a different engine in case they are not satisfied with the results from a search. Lastly, it has been experimentally attested [12] that searchers are generally unconcerned with specifying their preferred language of the documents retrieved by a search engine, since they assume that the language of their query will provide the needed selectivity. Therefore, users would issue more or less the same set of keywords for expressing an information need regardless of the query language, the engine's query handling capability or the language coverage of the engine's index.

With the above in mind and focusing on the Greek searcher, we have built a query selection mechanism that takes off the user the burden of specifying the query results' language and of trying to come up with keywords that the engine will understand for satisfying her information need. The details of our mechanism are presented next.

3.1 Processing Query Logs

The first step towards selecting good query alternatives is to be able to identify the latent user interests in their Web interactions. In other words, we need a way of computing the search profile of the user and then incorporate this profile in the query selection process. For implementing our profiling module, we have built a Web transactions monitoring plug-in, which resides at the client's side and records all the details of the user's interaction with a Web search engine. In particular, the data that our module collects concern the following: (i) the queries a user submits during a search session[5], (ii) the returned pages that the user visits for each of the queries and (iii) the frequency with which the same query is submitted within and across sessions.

[4] In case of a Greeklish query, we translate it in Greek.

[5] A search session is a time-contiguous sequence of queries issued by the same user [33].

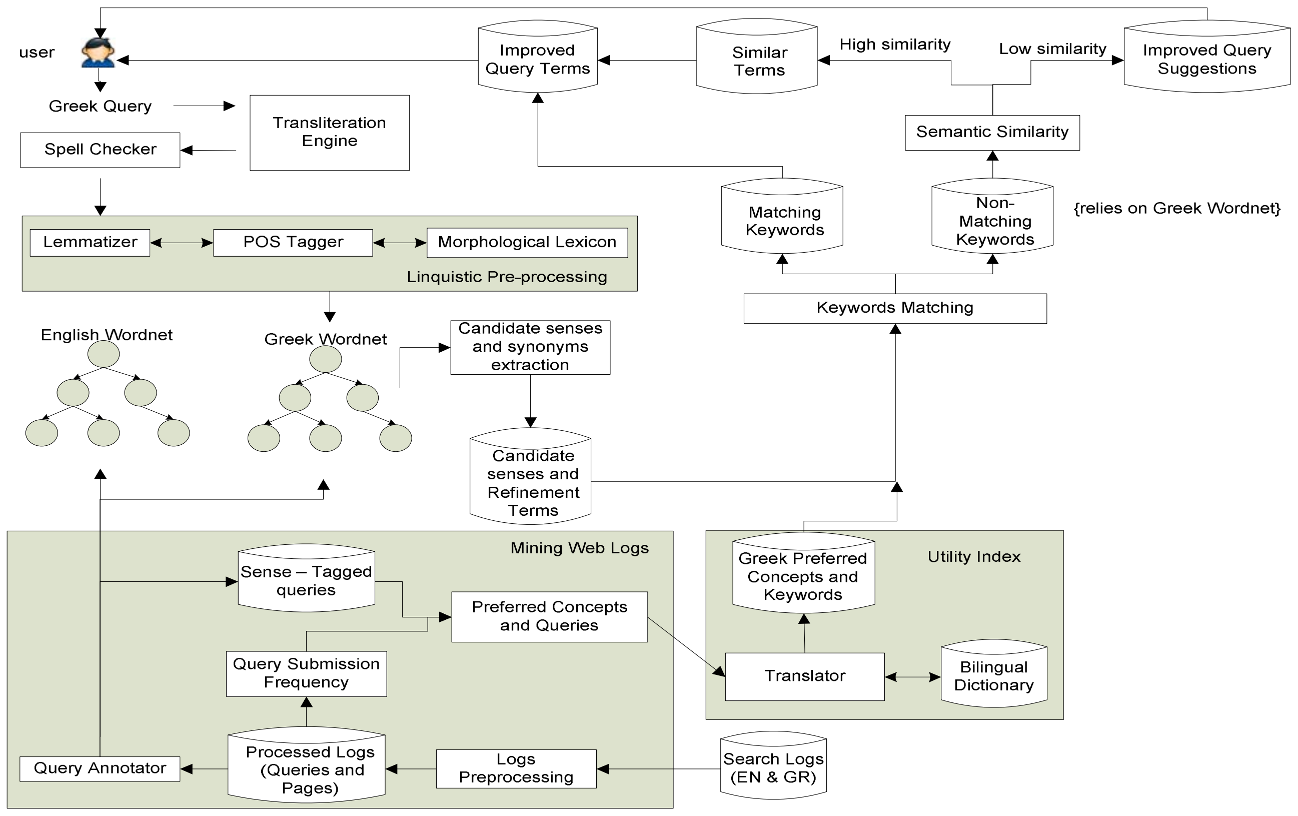


Figure 1. Query selection process.

Based on the data recorded in the user's search trace, we employ data mining techniques in order to derive the user interests and the user-preferred keywords for verbalizing those interests. Note that our plug-in module collects multilingual Web transaction logs, but here we focus our work on Greek and English searchers, since those two languages are mostly used by Greek Web users.

For estimating the user search interests, we explore the collected user's search trace, we clean it from noisy data and we group by query the pages that a user has visited in each session. Thereafter, we process the cleaned data in order to derive the query semantics. More specifically, we parse, tokenize, POS-tag and lemmatize the pages visited for every query. We remove their stopwords and we apply the TF*IDF weighting scheme [22] in order to assign an importance weight to the visited pages' content terms. We then take the n (n=25%) most highly weighted terms and we map them together with the query keywords to their corresponding WordNet nodes. Note that for Greek queries and visited pages, we rely on the Greek WordNet, whereas for English queries and pages we rely on the English WordNet [30]. We then apply the Wu and Palmer metric [31] in order to compute the semantic similarity between the query and the visited pages' important terms. According to Wu and Palmer, the similarity between two terms w_i and w_k is defined as:

$$\text{Similarity}\ (w_i, w_k) = \frac{2 * \text{depth}\ (\text{LCS}\ (i,k))}{\text{depth}(i) + \text{depth}(k)} \qquad (1)$$

Where LCS is the least common subsumer of the two terms. Based on the above formula, we pick the sense of the maximum similarity value and we assign it to the query.

Having annotated every past user query with an appropriate WordNet sense, the next step is to derive the user interests, i.e. the semantics of the queries that the user regularly submits in her Web searches. To account for the user interests, we again rely on the information collected from the user's search logs and we compute the degree to which a particular concept (i.e. sense) is preferred by the user. Formally, the degree of the user's interest in a particular concept is given by the frequency with which the user issues queries about that concept across her searches. To quantify that, we firstly compute the user interest in particular concepts within a session as:

$$\text{Session Interest}\ (C_i) = \frac{1}{|Q|} \sum_{C_i \in C}^{Q} \text{queries for}\ C_i \qquad (2)$$

Where Q is the total number of queries recorded in a single session and C is the set of concepts used to annotate the query semantics. Based on the above computations, we derive the user's general (i.e. across-session) interests as:

$$\text{User Interest}\ (C_i) = \frac{1}{|C|} \sum_{C_i \in C}^{C} \text{Session Interest}\ (C_i) \qquad (3)$$

The user interest values give us perceptible evidence about the degree to which particular concepts (i.e. topics) are generally preferred by the user in her Web searches. Based on the computed user interests and considering that all past user queries are annotated with an appropriate sense that expresses the identified interest, we can easily derive the set of keywords that the user prefers for expressing her conceptual interests. In particular, we rely on the collected search logs and we count the submission frequency of the queries annotated with the same concept. We then pick the m (m=10%) most frequently issued queries for every concept as the ones preferred by the user for expressing her interests.

Following the steps presented above, we can derive the user interests in her Web searches and the user-preferred keywords for expressing those interests. As a final step, we translate (where needed) all English queries and query selected senses to their Greek equivalents, via the use of a bilingual dictionary, and we store them locally in a utility index. Thereafter, we explore the data collected in the utility index in order to understand the semantics of the user's current query and improve it with alternative keywords.

3.2 Understanding and Improving Queries

Having computed the user's search profile (i.e. topical interests and preferred queries), we now turn our attention on how we utilize the above data for understanding the intention of the user's current query. Note that by current query, we mean the set of keywords that a user specifies in a new search, for which there is no clickthrough data.

Upon query issuing, the first step of our approach is to detect the query language. Considering that our mechanism has been designed for Greek language queries, we essentially need to ensure that the queries it processes are well formulated Greek queries. Thus, we rely on a language detection tool for identifying the language of the query and if the latter is written in Greeklish we pass it through our transliteration engine, which converts it to Greek. Thereafter, we apply spell-checking and we correct orthographic, intonation and spelling mistakes that might be either due to transliteration errors or due to the users' misspellings.

At the end of this process we come down to a set of correctly spelled keywords that the user defined for her new search. We then pass those keywords through a lemmatization engine, which induces every keyword to its first inflected form (i.e. lemma). The lemmatizer that we use in our work is the one described in [18] and it integrates a morphological lexicon and a Part-of-Speech tagger via which it resolves morphological ambiguities and detects the corresponding lemma of every wordform variant. Note that our lemmatizer is case sensitive and it achieves 98% accuracy in identifying the correct lemma for more than 1M Greek terms.

Following on, we map the lemmatized query terms to their corresponding Greek WordNet nodes and we extract all the possible senses that every word might have. In case none of the query terms is found in WordNet, the query is not further processed. For every extracted query sense, we also take its WordNet synonyms and we further process them in order to derive the query semantics. For query sense resolution, our method proceeds as follows. We rely on the data that represents the user's search profile, stored in the utility index against which we look for the query keywords. In case the exact keywords of the query are found in the index (i.e. the user has previously issued the very same query), we rely on the sense that has been appended to those keywords and we select it as the sense of the current query. Recall that past user queries are semantically annotated based on the semantics of the pages that the user has visited, i.e. pages on which the user has been active for more than 10 secs.. We then explore all the collected past user queries that have been annotated with the selected sense and upon their detection, we pick the keyword queries that are most preferred by the user and we suggest them as query alternatives. If there are no user preferred keywords for the selected query sense (i.e. the query has been previously issued only once by the user), we rely on the WordNet synonyms of the query matching sense and we use them as query alternatives.

On the other hand, if the current query is new, i.e. it has not been submitted before by the user, we attempt to understand and improve it as follows. We firstly look for semantically similar queries in the user's past search trace. In this respect, we apply the Wu and Palmer semantic similarity metric (cf. Section 3.1) and we compute how close are the candidate senses of the query to the senses that have been estimated for representing the user's search profile. In case a candidate query sense exhibits a high (i.e. above 0.85) semantic similarity to the senses that have been assigned to the user preferred concepts, we take the sense of the maximum similarity as the query sense and we improve the current query with the user preferred keywords associated with that sense. Again if there no user preferred keywords for the selected query sense, we use the WordNet synonyms of the query matching sense as query alternatives.

Lastly, in case none of the candidate query senses exhibits high semantic similarity to the concepts preferred in the user's past searches (i.e. if the query expresses a new user interest), we employ the WordNet synonyms for all the candidate query senses and we simply suggest them to the user so that she decides on which terms to select for improving her query. Note that in order to assist the user make an informed decision, our mechanism displays query suggestions accompanied by their corresponding WordNet sense. The final decision is then left on the user who can simply click on any of the system suggested terms and this is automatically appended to the initially typed keywords.

Following the process described above, we improve the user issued queries with alternative terms that are both expressive of the user's search interests and close to the current query semantics. In case the user's query represents a new search interest, we suggest to the user all possible alternative keywords and let her decide whether and how she will improve her search. We believe that our approach helps the user specify improved queries while at the same time she maintains control over her searches. To assess the usefulness of our query selection mechanism in a practical setting we have conducted an experiment, presented next.

4. EXPERIMENTAL EVALUATION

4.1 Experimental Setup

To evaluate the effectiveness of our query selection mechanism, we implemented a browser plug-in that records the users' search behavior and we asked five postgraduate students in our school install the plug-in for a period of three weeks and supply us with their search trace during that period. Note that the participants were not informed about the objectives or the nature of our study but they were simply told that we wanted to collect some information about how people interact with the Web. Recall that our plug-in records for every user her search sessions, the queries she issues in every session and the result pages on which she clicks for each of the queries. At the end of the recording period we collected a total number of 1,045 queries issued to Google[6] search engine, which span a number of 114 distinct sessions. The average number of queries issued in a single session is 9.1 and the average number of pages visited for a query is 3.9 (whereas the average number of viewed pages per query is 6.3). Of the 1.045 queries, 347 were unique, i.e. the have issued only once by a participant and the remaining 698 have been issued multiple times. Of those repeated queries, 124 have been issued more than once by the same user.

Having collected our experimental search trace, we employed our language detection module in order to identify the language of our experimental queries. Of the 1.045 queries, 734 were English language queries, 289 were Greek queries and the remaining 52 were Greeklish queries. The surprising observation that most of the collected queries are in English can be justified by the fact that all our participants are Computer Science students, who prefer to search in English for online data about their field of study.

To run our experiment, we relied on the collected data to which we performed a random 80/20 split. We then used the 836 queries (i.e. the 80% of the total queries) and the pages visited for those queries as our training set and the remaining 209 queries (i.e. the 20% of the total queries) and the pages visited for them as our testing set. Thereafter, we processed each of the queries and the query visited pages in our training set following the steps described in Section 3.1, in order to derive for every user the set of concepts that represent her search interests and the set of keywords that the user prefers for expressing those interests. Note that queries and pages in the training set are grouped by user. At the end this process, we came down to an average set of 4.6 interesting concepts (i.e. topics) per user and 8.3 preferred keywords per topic.

Based on the computed user interests and interest-expressive keyword preferences, we evaluated our model's effectiveness in exploring the above knowledge towards selecting good query alternatives. Before starting our evaluation, we examined the language of our testing queries in order to ensure that our study concentrates on Greek queries. Out of the 209 testing queries, 70 were initially written in either Greek or Greeklish. Therefore, we relied on these 70 queries and we supplied them together with the user profiles computed in the training set to our query selection mechanism. Our module processed each of the queries as presented in Section 3.2; that is, it corrected misspellings, it transliterated Greeklish keywords to Greek, it lemmatized query terms and looked them up in the Greek WordNet in order to extract their candidate senses and sense synonyms. Afterwards, and relying on the computed user interests and preferred keywords, our module selected for each of the 70 test queries alternative terms, following the steps previously described. At the end of the query selection process, our module delivered on average 4.7 alternative terms for each of the testing queries. Note that alternative terms are separated by spaces in order to formulate the improved query, which is executed as a Boolean OR query.

For our evaluation, we issued the improved queries that our model delivered to Google search engine and we compared their effectiveness in delivering qualitative results to the effectiveness that the same queries had before their improvement. In our evaluation, we perceive the results returned by the user issued queries before these have been processed by our method as baseline retrieval performance, while we consider the results returned by the queries selected by our model for the user typed keywords as improved retrieval performance. To quantify the improved queries' effectiveness we examined whether the most interesting page for each of the queries before their improvement is ranked among the first three positions in the results of the corresponding improved queries. In our assessment, the most interesting page for a query is the page that has received most of the user clicks for that query across its multiple submissions and on which the user has been active for a long period of time, typically longer than the average duration of that user's page visits. Therefore, query interesting pages are estimated regardless of their rank position in the search results, diminishing thus any potential search engine bias to the results' quality.

Apart from the above experiment, we also measured the usefulness of the alternative keywords that our model selects in retrieval performance. In particular, we computed the amount of the system selected terms that participate in the improved searches in order to assess how many terms should participate in a refined query, so that the latter is not too generic or overly specific. Next we report obtained results.

4.2 Experimental Results

Table 1 presents the success rate of our experimental queries in delivering highly interesting search results before and after their improvement. In particular, the Table shows the number of queries (as these have been defined by the user (i.e. baseline) and as these have been improved by our selection module) that returned the most interesting query page in the first three rank positions, as well as the number of queries (before and after improvement) that failed to retrieve the most interesting page for the query in the first three ranking positions.

Table 1. Performance of improved queries

Interesting Page Ordering	**# Successful Queries**		**Queries' success rate**		**Retrieval Improvement**
	User-Defined	**System-Selected**	**User-Defined**	**System-Selected**	**System Selected Queries**
Rank 1	14	22	20%	31.4%	36.3%
Rank 2	12	19	17.2%	27.2%	36.7%
Rank 3	17	18	24.3%	25.7%	5.4%
			Queries' failure rate		
Missed Pages (below Rank 3)	27	11	38.5%	15.7%	N/A

Of the interesting pages identified it appears that most of them were ordered in a high position (i.e. up to rank level 3) by an improved query that our model selected. The improved queries have higher success rates ranging from 31.4% to 25.7% in delivering qualitative results than the user defined ones, whose success rate

[6] We restricted our experimental data to queries submitted to Google search engine not only because it is the most popular engine for Greek searchers but also because it is easier to collect the queries headed against it based on the syntax of the URL in the address bar.

range between 24.3% and 17.2%. Moreover from the results reported in Table 1, we can see that our query selection mechanism yields an overall retrieval improvement of 78.4% at rank position 3 compared to the retrieval performance of the corresponding user typed queries at the same rank position. Although not reported due to space considerations, a detailed analysis of our results indicates that our model's performance is not significantly affected by the nature of the individual queries, i.e. the number of keywords they initially contain or the number of similar terms previously issued by the user. What has an impact though on the performance of our system is the amount of training data needed for identifying the preferred query concepts and concept-related keywords. In particular, we observed that as the size of the training data (i.e. pages visited for some query) increases so does the ability of our model in selecting good query alternatives. In a current study, we are experimenting with the minimum amount of clickthrough data required by our model in order to operate in an effective yet efficient manner.

The results reported so far give an overall performance rate for our query selection mechanism compared to baseline retrieval (i.e. user issued queries), but they do not show how many of the system selected keywords actually participate in the improved searches. Table 2 presents the selected keywords' distribution in the first three retrieved pages for our improved queries. The Table shows the number of the query alternative keywords that appear in the contents of the improved search results at recall level 3. For instance, we can see that out of the 22 most interesting pages that are ranked first in the improved query results, 4 of them contain all the query alternatives that our model selects, 15 contain some of the terms (but not just one) that our model selects and 3 pages contain only one of the system selected terms. Note that in the above estimations we have excluded the user typed keywords but rather concentrated only on system selected terms.

Table 2. Selected terms' distribution in search results

# of selected query terms in search results	Pages containing alternative keywords		
	Rank 1	Rank 2	Rank 3
All	4	2	0
Some but not one	15	12	8
One	3	5	10

Based on the results reported in Table 2 and considering that the average number of keywords that our model selected for improving our experimental queries is 4.7, we may conclude that a number of 3 to 4 alternative keywords suffice for retrieving qualitative search results. An implicit finding is that executing the refined queries as simple Boolean OR requests is better than applying advanced search options. Considering that users rarely define complex query formulations, we believe that our query selection mechanism complies with the querying patterns that the majority of Web users employ.

In overall, experimental results demonstrate that our query selection mechanism has a significant potential in improving Greek Web searches. Although further experimentation is needed before we fully deploy our method, we believe that our preliminary findings will pave the way for further studies (hopefully in other languages) on the impact that past multilingual user searches might have in selecting good search keywords in non-English languages.

5. CONCLUDING REMARKS

In this paper, we introduced a novel query selection mechanism for improving Greek Web searchers. Our proposed mechanism relies on the analysis of the user's past search behavior in order to derive both the user interests and the user preferred keywords for expressing those interests. Following on from that, it examines the user's new queries, for which there is no click data available and selects alternative words for their improvement. Query selection is greatly dependent upon the linguistic and the semantic processing of the terms in both the query and the user interesting pages. A preliminary evaluation of our query selection mechanism demonstrates the potential of our method in improving Greek Web searches. One innovative aspect of our work is that we rely on multilingual Web transaction logs for deriving the user search interests. Thus, unlike previous attempts that mine Web logs in a single language in order to learn the user profiles, we explore the entire search trace of the user regardless of the natural language in which the query or the query results are written.

Another innovative aspect of our query selection mechanism is that it integrates into a single module all the different strategies proposed in the literature for improving search queries. In particular, our method selects alternative queries based on: (i) feedback terms [23] i.e., terms that the user prefers in her searches, (ii) dependent terms [20], i.e. terms that are similar to the query keywords, and (iii) it employs interactive selection [11], i.e. it computes a list of candidate terms, presents them to the user and lets the latter decide which terms are to be included in the refined query.

Our plans for future work concentrate on, but they are not limited to, the following issues. We are currently underway of a large scale evaluation experiment in which we are testing the learning accuracy of our user profiling module. For this study, we are using a much larger sample of transaction logs and we apply a number of heuristics for fine-tuning our method's learning ability. Next, we plan to assess the effectiveness of our query selection module towards a multitude of different query types issued to different search engines (both global and local). We expect that such an experiment will shed some light on the different improvement techniques that should be applied to different query types and it will also help us decipher any potential underlying association between user interests and query types. Lastly, it would be interesting to investigate whether the alternative queries that our system selects should be suggested to the user in a particular order or following a specific syntax in order to ensure that they are properly perceived by the searcher. Above all, it is our hope that our study will stimulate the interest of other researchers into advancing our mechanism for improving Web searches in languages other than English.

6. REFERENCES

[1] Baeza-Yates, R., Castillo, C. and Efthimiadis, E. 2004. Comparing the characteristics of the Chilean and the Greek Web. Technical Report http://citeseer.ist.psu.edu/680836.html

[2] Baeza-Yates, R., Castillo, C. and Efthimiadis, E. 2007. Characterization of national Web domains. In ACM Transactions on Internet Technology, 7(2), Article 9.

[3] Bar-Ilan, J. and Gutman, T. 2005. How do search engines respond to some non-English queries? Journal of Information Science, 31(1): 13-28.

[4] Billerbeck, B., Scholer, F., Williams, H.E. and Zobel L. 2003. Query expansion using associated queries. In Proceedings of the ACM CIKM Conference.

[5] Chan, M., Fang, X. and Yang, C.C. 2007. Web searching in Chinese: a study of a search engine in Hong Kong. Journal of the American Society for Information Science and Technology, 58(7): 1004-1054.

[6] DeLuca, E.W. and Nurnberger, A. 2006. Adaptive support for cross-language text retrieval. In Proceedings of the Intl. Conference in Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 425-429.

[7] Efthimiadis, E., Malevris, N., Kousaridas, A., Lepeniotou, A. and Loutas, N. 2008. How search engines respond to Greek language queries. In Proceedings of the IEEE Intl. Conference on System Sciences, Hawaii.

[8] Grigoriadou, M., Kornilakis, H., Galiotou E., Stamou, S. and Papakitsos, E. 2004. The software infrastructure for the development and validation of the Greek WordNet. In Romanian Journal of Information Science and Technology, 7(1-2).

[9] Gong, Z., Cheang, C.W. and Hou, C. 2005. Web query expansion by WordNet. In Proceedings of the DEXA Conference.

[10] Gey, F.C., Kando, N. and Peters, C. 2005. Cross-language information retrieval: the way ahead. Information Processing and Management, 41(3): 415-431.

[11] Harman, D. 1988. Towards interactive query expansion. In Proceedings of the 11th Intl. Conference on Research and Development in Information Retrieval, pp. 321-331.

[12] Jansen, B.J. and Spink, A. 2005. An analysis of Web searching by European AlltheWeb.com users. In Information Processing and Management, vol.41, pp. 361-381.

[13] Mchedlidze, T., Symvonis, A. and Tzagarakis, M. 2007. Analysis of the Greek Web-space. In Proceedings of the 11th Panhellenic Conference on Informatics, Patras, Greece.

[14] Moukhad, H. and Large, A. 2001. Information retrieval from full-text Arabic databases: can search engines designed for English do the job? In Libri, pp. 63-74.

[15] Neumann, G. and Xu, F. 2003. Mining answers in German Web pages. In Proceedings of the Conference on Web Intelligence.

[16] Lampos, Ch., Eirinaki, M., Jevtuchova, D. and Vazirgiannis, M. 2004. Archiving the Greek Web. In Proceedings of the 4th Intl. Web Archiving Workshop, Bath, UK.

[17] Lazarinis, F. Web retrieval systems and the Greek language: Do they have an understanding?. 2007. In Journal of Information Science, pp. 1-15.

[18] Ntoulas, A., Stamou, S. and Tzagarakis, M. 2001. Using a WWW search engine to evaluate normalization performance in a highly inflectional language. In Proceedings of the ACL/EACL Student Research Workshop, France.

[19] Pew Internet &American Life Project. 2005. Search engine users: Internet searchers are confident, satisfied and trusting– but they are also unaware and naïve. Report a available at: http://www.pewinternet.org/pdfs/PIP_Searchengine_users.pdf

[20] Pirkola, A. 1999. Studies on linguistic problems and methods in text retrieval: the effects of anaphor and ellipsis resolution in proximity searching and translation and query structuring methods in cross-language retrieval. Ph.D. Dissertation

[21] Qin, J., Zhou, Y., Chan, M. and Chen, H. 2003. Supporting multilingual information retrieval in Web applications: an English-Chinese Web portal experiment. In proceedings of the 6th Intl. Conference on Asian Digital Libraries, pp.149-152.

[22] Salton, G. and McGill, M.J. 1983. Introduction to Modern Information Retrieval. McGraw-Hill, ISBN: 0070544840.

[23] Salton, G. and Buckley, C. 1990. Improving retrieval performance by relevance feedback. In Journal of the American Society for Information Science, 41(4), pp. 288-297.

[24] Spink, A., Ozmultu, S., Ozmultu, H.C. and Jansen, B.J. 2002. U.S. versus European Web searching trends. In ACM SIGIR Forum, vol.15. no.2.

[25] Spink, A. 2003. Web search: Emerging patterns. In Library Trends, vol. 52, no.2, pp. 299-306.

[26] Sroka, M. 2000. Web search engines for Polish information retrieval: questions of search capabilities and retrieval performance. Information and Library Research, 32.

[27] Stamou, S. and Christodoulakis, D. 2005. Retrieval efficiency of normalized query expansion. In Proceedings of the 6th Intl. Conference on Computational Linguistics and Intelligent Text Processing. Mexico, pp. 604-607.

[28] Tzekou, P., Stamou, S., Zotos, N. and Kozanidis, L. 2007. Querying the Greek Web in Greeklish. In Proceedings of the SIGIR Workshop on Improving non-English Web Searching, Amsterdam, the Netherlands.

[29] Tzekou, P., Kozanidis, L., Christodoulakis, D. and Stamou, S. 2006. Combining statistical and lexical processing for query refinement. In Proceedings of the 5th Intl. Conference on Formal Approaches to South Slavic & Balkan Languages.

[30] WordNet: hhttp://wordnet.princeton.edu.

[31] Wu, Z. and Palmer, M. 1998. Web semantics and lexical selection. In Proceedings of the 32nd ACL Meeting.

[32] Smyth, B., Balfe, E., Freyne, J., Briggs, P., Coyle, M. and Boydell, D. 2005. Exploiting query repetition and regularity in an adaptive community-based web search engine. In User Modeling and User Adapted Interaction, 14(5): 383-423.

[33] Kammenhuber, N., Luxenburger, J., Feldmann, A. and Weikum, G. 2006. Web search clickstreams. In Proceedings of the 6th ACM SIGCOMM Conference on Intern Measurement, pp. 245-250.

A Study of Using an Out-Of-Box Commercial MT System for Query Translation in CLIR

Dan Wu
School of Information Management
Wuhan University, Hubei, China
woodan@whu.edu.cn

Daqing He
School of Information Sciences
University of Pittsburgh, PA, USA
dah44@pitt.edu

Heng Ji, Ralph Grishman
Computer Science Department
New York University, NY, USA
{hengji, grishman}@cs.nyu.ed

ABSTRACT
Recent availability of commercial online machine translation (MT) systems makes it possible for layman Web users to utilize the MT capability for cross-language information retrieval (CLIR). To study the effectiveness of using MT for query translation, we conducted a set of experiments using Google Translate, an online MT system provided by Google, for translating queries in CLIR. The experiments show that MT is an excellent tool for the query translation task, and with the help of relevance feedback, it can achieve significant improvement over the monolingual baseline. The MT based query translation not only works for long queries, but is also effective for the short Web queries.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Query formulation

General Terms
Experimentation, Languages, Performance

Keywords
Cross-Language Information Retrieval, Query Translation, Machine Translation, Query Expansion, Relevance Feedback

1. MT FOR QUERY TRANSLATION IN CLIR

Recent rapid development of new Internet technologies has empowered more people to use the Web as their social and collaboration platform. As every Web user potentially becomes not only a Web information consumer but also an information creator, cross-language information retrieval (CLIR) becomes critical for the communication on the Web. Considering most Web users are non-native English speakers, and most Web information is in English, CLIR is a critical Web search service for non-English searchers.

CLIR often relies on translation methods to cross the language barriers between a query and the documents. Depends on whether it is the query, the documents, or both that are translated, we have query translation based CLIR, document translation based CLIR and interlingual CLIR. Many resources have been exploited for the translation task, among which the most commonly used are machine-readable dictionary (MRD), parallel or comparable corpora, and machine translation (MT) systems. MRD is probably the most commonly used translation resources in experiment setting, especially for translating queries [3, 6]. However, if consider the easy accessibility for some popular languages, MT is probably the most commonly available translation resource among the above three on the Web. Several companies such as Google and Alta Vista are actively prompting their multilingual services on the Web. Therefore, it is reasonable to assume that a layman Web user can potentially achieve a cross-language information retrieval by using one of the commercial online MT services to translate his/her queries.

Although MT in document translation based CLIR can generate competitive search results to monolingual searches, in the context of query translation based CLIR, the effectiveness of using MT to translate queries is much more uncertain in previous studies. On the one hand, researchers criticized that the quality of MT translation was poor [7], and dictionary-based CLIR techniques are often cited to outperform those using some popular commercial MT for translating queries [4]. On the other hand, Oard find that early rule-based MT systems when were used for translating queries can generate comparable or slightly better CLIR results than dictionary-based methods [10]. Other researchers also showed that MT-based query translation can achieve reasonable effectiveness, and was better than the approach that using all the translations in the dictionary [1].

However, we have seen two important developments in the fields of CLIR and MT in recent years. First, both MT and CLIR have experienced rapid development of integrating statistical based language models and resources into their handling of translations. Statistical MT has becomes the state of the art for MT [11], and even some commercial MT systems such as Google Translate are statistical MT systems. Translation probabilities are widely used in CLIR for handling translation ambiguities or are even built as a part of the statistical language modeling for CLIR [6, 9, 14]. One important insight gained in CLIR from the usage of translation probabilities is that choosing multiple possible translations with their probabilities is a superior method than choosing only the top best translation. This insight argues against of the usage of MT output, which is the list of one best translation for each query term, as the translation of the query.

Second, both MT and CLIR have to face out-of-vocabulary terms (e.g., those terms whose translations are not available in the translation resources such as MRD). In CLIR, recent studies show that it is beneficial to use dedicated data mining and information extraction methods for obtaining high quality translations for named entities [8], which are the most important and most common type of OOV [12]. This again can potentially help MRD method to outperform MT based query translation.

Therefore, with all these recent developments in both MT and CLIR, we are interested in examining again the effect of using MT in query translation based CLIR. Our work has a great benefit of answering the question that, by using current MT systems as their query translation resource, can laymen searchers obtain CLIR performance that is comparable to that of using MRD. We will concentrate on using commercial MT systems such as Google Translate since they are the ones that most people can access and integrate into their CLIR process. To cover multiple CLIR scenarios, our research questions cover three important factors:

- Do pseudo relevance feedback (PRF) and the different CLIR query expansion (QE) methods make difference to the usage of MT and MRD in query translation? In CLIR, there are pre-translation QE that performed before translating the query with the help of an extra document collection at the query language side, post-translation QE that performed after the query is translated, and the combination of pre and post-translation QE, which is called combined QE. The motivation here is that PRF has demonstrated to gain significant improvement on retrieval effectiveness in both monolingual IR and CLIR, and different QE methods seem work differently.
- Do different query lengths make difference? Web search queries are often short, but queries in TREC like evaluation frameworks are often much longer. We, therefore, utilize the title, description, and narrative fields in the search topics to simulate short Web queries and long TREC queries, and examine the differences between query translation based on MT and that based on MRD.
- Do the handling of out-of-vocabulary (OOV) terms by dedicated NE translation component change the results obtained in the two previous research questions?

2. EXPERIMENT SETTINGS

Our experiments were performed between English queries and Chinese documents. The reasons for this selection include that CLIR between English and Chinese is still a very active research topic, and all the authors have been working on Chinese related issues.

2.1 CLIR Systems in the Experiments

At MT side, we selected Google Translate[1] as the out-of-box commercial statistical MT for translating queries. Google Translate is a Web multilingual service provided by Google Inc. The underneath MT system is a statistical based system, which was developed by Franz-Josef Och, and the system won the DARPA contest for speed machine translation in 2003[2]. We did not and could not modify any parameter of Google Translate. Although this prevents us from studying Google Translate in depth, this setting is actually close to what would happen when layman Web users incorporate Google Translate into their CLIR search. When using MT for translation, rather than translate the query terms one by one, we inputted the whole query into the MT system at once.

The bilingual MRD used for our dictionary-based CLIR was an English-Chinese lexicon generated from a parallel bilingual corpus automatically [13]. It was compiled by training GIZA++[3] on multiple sources including the Foreign Broadcast Information Service (FBIS) corpus, HK News and HK Law, UN corpus, and Sinorama, et al. The dictionary contains 126,320 English entries with translation probabilities for each Chinese translation alternative. The translation probabilities are obtained based on the normalized frequency of an English word and a Chinese word being linked together by word alignment.

During the translation of the queries with the MRD, to remove low probability translations which often are noises, a fixed threshold called Cumulative Probability Threshold (CPT) is selected. This is done by ranking the translations in decreasing order of their probabilities, then iteratively selecting translations from the top of the ranking until the cumulative probability of the selected translations firstly reaches or exceeds the threshold. A threshold of 0 thus corresponds to the using the single most probable translation (a well-studied baseline) and a threshold of 1 corresponds to the use of all translation alternatives in the dictionary.

In order to improve the coverage of the dictionary as much as possible, we adopted the back-off translation strategy [5] during the translation of the query terms. The back-off strategy starts with trying to match the surface form of an input English term to the surface forms of English term in the dictionary first, if it fails, stem the input English term and match the stem of the input term to surface forms of English term in the dictionary, if this still fails, stem the dictionary and match the surface form of the input term to stems of terms in the dictionary, if all else fails, match the stem of the input term to the stem of terms in the dictionary.

Once the queries were translated by either MT or MRD, we used Indri v.2.4[4] as the search engine to perform document retrieval on the Chinese collection.

For all the PRF tasks in the experiments, we used the Indri's build-in PRF module which is based on Lavrenko's relevance model [9] . Formula (1)[5] summarizes this implementation, where I is the original query, r is a term to be considered for query expansion, and D is a document.

$$P(r \mid I) = \frac{\sum_{D} P(r \mid D) P(I \mid D) P(D)}{P(I)} \qquad (1)$$

[1] http://translate.google.com/translate_t?hl=en

[2] Based on http://en.wikipedia.org/wiki/Google_Translate

[3] http://www-i6.informatik.rwthaachen.de/Colleagues/och/software/GIZA++.html/

[4] http://sourceforge.net/projects/lemur/

[5] http://ciir.cs.umass.edu/~metzler/ indriretmodel.html

The parameters used in our PRF tasks were selecting top 20 terms from top 20 returned documents. The relative weight between original queries terms and expanded queries is 0.5. This was based on our previous ad hoc exploration of the parameters in Indri.

2.2 Information Extraction-based Named Entity Translation Module

In MRD based CLIR, we adopted an NE translation component based on information extraction (IE) techniques. The IE component is designed to provide two functions in the MRD based query translation. The first one is to identify named entities in a given text, which could be queries, documents, or any parts of queries and documents. The function was provided by the NYU English and Chinese HMM-based name taggers trained on several years of ACE (Automatic Content Extraction[6]) corpora. Both name taggers can identify seven types of names (Person, Geo-Political Entity (GPE), Location, Organization, Facility, Weapon and Vehicle) and achieve about 87%-90% F-measure on newswire [7].

The English HMM NE tagger [7] includes six states for each main name type, as well as a not-a-name state. These six states correspond to the token preceding the name; the single name token (for names with only one token); the first token of the name; an internal token of the name (neither first nor last); the last token of the name; and the token following the name. These multiple states allow the HMM to capture context information and the information about the internal structure of the name.

The Chinese name tagger consists of a HMM tagger augmented with a set of post-processing rules. The HMM tagger generally follows the Nymble model [5]. Within each of the name class states, a statistical bigram model is employed, with the usual one-word-per-state emission. The various probabilities involve word co-occurrence, word features, and class probabilities. Since these probabilities are estimated based on observations seen in a corpus, several levels of "back-off models" are used to reflect the strength of support for a given statistic, including a back-off from words to word features, as for the Nymble system. To take advantages of Chinese names, we extend a model to include a larger number of states, 14 in total. The expanded HMM can handle name prefixes and suffixes, and has separate states for transliterated foreign names. Finally a set of post-processing heuristic rules are applied to correct some omissions and systematic errors.

The second function of the IE component is to handle the translations of identified NEs. Our IE component exploits the NYU name translation system [8], which usages the following methods to locate possible translations from a variety of resources:

- *Extracting cross-lingual name titles from Wikipedia pages*. We run a web browser [2] to extract titles from Chinese Wikipedia pages and their corresponding linked English pages (if the link exists). Then we apply heuristic rules based on Chinese name structure to detect name pairs, for example, foreign full names must include a dot separator, Chinese person names must include a last name from a close set of 437 family names.
- *Tagging NEs in parallel corpora*. Within each sentence pair in a parallel corpus, we run the NE taggers on both sides. If the types of the NE tags at both sides are identical, we extract the NE pairs from this sentence. Then at the corpus-wide level, we count the frequency for each NE pair, and only keep the NE pairs that are frequent enough. The NE pair then becomes the translation of each other. The corpora used for this approach were all GALE[7] MT training corpora and ACE 07 Entity Translation training corpora to conduct this process. We didn't use word alignment information because the state-of-the-art Chinese-English word alignment is only about 60%.
- *Using patterns for Web mining*: we constructed heuristic patterns such as "Chinese name (English name)" to extract NE pairs from web pages mixed of Chinese and English.

Due to different sources, usages, and preferences, the same NE may have different translations. Some of them are all correct but with different transliteration schemes. For example, the person name "Albright" can have Chinese translation "奥尔布赖特" or "阿尔布赖特" according to different schemes. The organization name "UNESCO" has its Chinese translation "联合国教科文组织" or just "教科文组织" according to its different abbreviations. For these types of translation alternatives, it actually more important that possible translations are all there. It is less important to have a weight to differentiate their importance. However, the weights are essential to handle the following situation. Some translations are errors generated through the automatic IE process. If without any further indication, the retrieval system would treat them equally important to the correct translations. This error would reduce the retrieval effectiveness. To solve this problem, we utilized a popular commercial Web search engine[8] to obtain three numbers, which correspond to the numbers of web sites contain the NEs, the translation, and the NE and the translation together in the same page:

$$weight(NE_i, tran_{i,j}) = \alpha \frac{|NE_i \cap tran_{i,j}|}{|NE_i| + |tran_{i,j}|} \qquad (2)$$

where $weight(NE_i,tran_{ij})$ is the weight for translation j of NE i, $|NE_i,tran_{ij}|$ is the number of returned pages containing both NE_i and $tran_{ij;}$ $|NE_i|$ is the number of returned pages containing NE_i and $|tran_{ij}|$ is the number of returned pages containing $tran_{ij;,}$ α is a constant to adjust the score in case the weight is too small.

After getting all the weights for each NE_i, we normalize the sum of $weight(NE_i,tran_{ij})$ to 1:

$$\sum_{j=1}^{n} weight^{'}(NE_i, tran_{ij}) = 1 \qquad (3)$$

where n is the total number translations of NE_i.

Just like that there are OOV terms when using normal dictionary for query translations, there are OOV NEs when using our IE generated NE resources. To fix this problem, we developed two simple patterns that look for the English NEs either appear in

[6] http://www.nist.gov/speech/tests/ace/

[7] Global Autonomous Language Exploitation

[8] http://www.google.com

brackets after a Chinese word (i.e., … Chinese word (English NE)…) or a Chinese word in brackets after the English NE (i.e., … English NE (Chinese Word)…). The motivation is that many Chinese web pages would give the English or Chinese translation when a new NE is introduced. Our search was performed on baidu.com[9] which is the most popular Chinese search engine.

2.3 Search Collections and Topics

The test collection used in our study combined TDT4 and TDT5 Multilingual News Text corpora. The combined collection contains 83,627 Chinese documents of 328M and 306,498 English documents of 1.13G. The collection was used for NIST multilingual Topic Detection and Tracking evaluations [reference].

We selected 44 TDT English topics and manually translated them into Chinese for monolingual Chinese search. We also rewrote them into TREC topic style with title, description and narrative fields (see Figure 1). Queries were automatically extracted from the topics with short queries containing titles only (T query), medium queries with title and description fields (TD query), and long queries with all the three fields (TDN query). The average length of the queries were: T query (4 terms), TD query (27 terms), and TDN query (127 terms).

<num> Number: 41012
<E-title> Trouble in the Ivory Coast

<E-desc> Description:
Presidential election; Laurent Gbagbo, Alassane Ouattara, Ivory Coast voters; Ivory Coast; October 25, 2000

<E-narr> Narrative:
On October 25, Laurent Gbagbo, head of the Ivorian Popular Front, declared himself president, as early polls showed him in the lead. Alassane Ouattara called the election unfair, but then conceded, though tens of thousands of his supporters took to the streets. A recent history of power struggle that led to the current election. Disputes concerning the election including violence by the opposition groups.

Figure 1: An example of the modified TDT topic

All Chinese texts and queries were segmented using the Stanford Chinese word segmenter[10]. The Porter stemmer[11] was used to stem English texts, queries and the dictionary when necessary. Stop words were removed using a Chinese stopword list[12] and an English stopword list[13]. After all these preprocessing steps, the queries are ready for translation by either MT or MRD,

To answer our research questions, we conducted the following experiment runs:

- Monolingual Baseline: a monolingual run that retrieves Chinese documents using the manually translated Chinese topics.
- MT CLIR Baseline: a CLIR run that translates English topics with MT.
- MT QE-PreTrans: a CLIR run that uses MT to translate queries and uses the pre-translation QE method for PRF.
- MT QE-PostTrans: a CLIR run that uses MT to translate query and uses the post-translation QE method for PRF.
- MT QE-Combine: a CLIR run that uses MT to translate query and uses the combination of pre-translation and post-translation QE for PRF.
- MRD CLIR Baseline: a CLIR run that translates English topics using the MRD.
- MRD NE Baseline: a CLIR run that translates queries using the MRD and translates NEs using the NE module.
- MRD QE-PreTrans: a CLIR run that uses the MRD to translate the queries and uses the pre-translation QE method for PRF.
- MRD QE-PostTrans: a CLIR run that uses the MRD to translate the queries and uses the post-translation QE method for PRF.
- MRD QE-Combine: a CLIR run that uses the MRD to translate the queries and uses the combined QE method for PRF.

Figure 2 illustrates the experiment procedure involving MT or MRD based query translation with and without PRF.

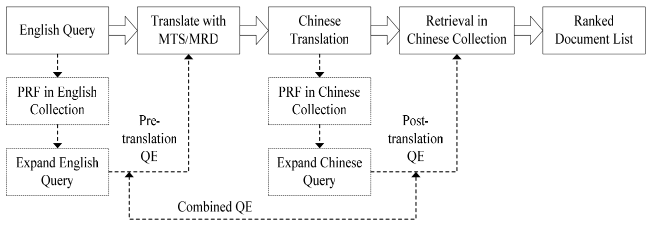


Figure 2: The whole experiment procedure

When MT is used for query translation, it only outputs the one best translation for each query term. However, for MRD runs, we tried different CPTs from 0.0 to 1.0 with an increment of 0.1 at each time. The results presented here are the best run results of the 11 CPTs (CPT=0.5 for all three query lengths).

We use mean average precision (MAP) as the basis of evaluation for all experiments and the two tailed paired samples t-test at P-value < 0.05 for statistical significance test.

[9] http://www.baidu.com

[10] http://nlp.stanford.edu/software/segmenter.shtml

[11] http://tartarus.org/~martin/PorterStemmer/

[12] http://www.unine.ch/info/clef/englishST.txt

[13] http://bbs.ir-lab.org/cgi-bin/topic.cgi?forum=3&topic=127

3. RESULTS AND DISCUSSION

3.1 MT-based Query Translation in CLIR without PRF

Table 1 shows the MAP results of all the experiment runs. For our discussion, we first examine the runs without PRF, which include MT CLIR Baseline, MRD CLIR Baseline, MRD NE Baseline and Monolingual IR.

Without any query expansion improvement from feedback, most CLIR runs could not generate comparable results to Monolingual IR, particular MRD based query translation method, no matter using or not using NE translation module, and whether the queries are T, TD or TDN.

Table 1: The Mean Average Precision (MAP) results of MT-based and MRD-based CLIR runs

Run ID	T	TD	TDN
Monolingual IR	0.4739	0.5817	0.6215
MT CLIR Baseline	0.4446	0.5536	0.6170
MT QE-PreTrans	0.4922	0.5443	0.5580
MT QE-PostTrans	0.5284	0.6031	0.6292
MT QE-Combine	0.5604	0.5833	0.6001
MRD CLIR Baseline	0.3336	0.4251	0.4701
MRD NE Baseline	0.3934	0.5034	0.5563
MRD QE-PreTrans	0.3714	0.4377	0.4477
MRD QE-PostTrans	0.4118	0.5080	0.5182
MRD QE-Combine	0.4415	0.5007	0.5170

However, MT based query translation method under TDN queries did generate comparable results to Monolingual IR (99% of Monolingual IR). The performance of MT based query translation was much closer to Monolingual IR under T and TD queries too (at least 94% of Monolingual IR). This indicate that it is clear that MT based query translation is a better query translation method than MRD based method when there is no query expansion.

Table 2: Significant T-test on Comparison (* indicates that the improvement is statistically significant)

	Impr. over MRD CLIR Baseline	Impr. over MRD NE Baseline
MT CLIR Baseline (T)	+33.27%*	+13.01%
MT CLIR Baseline (TD)	+30.23%*	+9.97%
MT CLIR Baseline (TDN)	+14.69%*	+10.91%*

Table 2 shows the significant testing between the runs of MT against that of MRD. MT-based query translation significantly outperformed MRD CLIR Baseline in all three lengths of queries. However, it is interesting to see that the superiority of MT method is shrinking along with the increasing of query length. This is counter intuitive since we would think that MT performs better with more context (longer queries). We will conduct further analysis on this issue.

As shown in Tables 1 and 2, introducing NE translation module does improve the performance. Although MRD NE Baseline is still inferior to that of MT CLIR Baseline, the difference is not significant in two of the three query lengths (T and TD).

3.2 MT-based Query Translation in CLIR with PRF

When PRF was added into MT-based query translation CLIR, all runs were at least 90% of monolingual performance, with many of them over 100% and the highest is 120% (MT QE-Combine at T queries). Therefore, all these MT-based CLIR run have achieved the state of the art CLIR performance which is either comparable to or higher than the performance of corresponding monolingual run. It is interesting to notice that MT works even with short queries (Query T). Lacking of context was a worry about using MT for query translation. All these results confirm again that MT is a valid tool for translating queries in CLIR.

Table 3: Comparison of QE methods for MT-based CLIR (* indicates the improvement is statistically significant)

		Perc. Of Mono IR	Impr. over MT-based CLIR
T	MT QE-PreTrans	103.86%	10.71%*
	MT QE-PostTrans	111.50%*	18.85%*
	MT QE-Combine	118.25%*	26.05%*
TD	MT QE-PreTrans	93.57%	-1.68%
	MT QE-PostTrans	103.68%	8.94%*
	MT QE-Combine	100.28%	5.36%
TDN	MT QE-PreTrans	89.78%*	-9.56%*
	MT QE-PostTrans	101.24%	1.98%
	MT QE-Combine	96.56%	-2.74%

As shown in Table 3, although in general query expansion (QE) based on PRF helps MT based query translation CLIR, its effect is affected by the query lengths. QE was most helpful when queries are short. Under T queries, all three QE methods significantly outperformed not only the MT CLIR Baseline, but also the Monolingual Baseline. The post-translation QE and the combined QE runs even significantly outperformed the monolingual baseline. However, when queries are longer, like in the cases of medium TD queries and long TDN queries, the effectiveness of QE is shrinking. Not all QE methods generate significant improvement over even the CLIR baseline. In fact, several runs (MT QE-PreTrans under TD queries, MT QE-PreTrans and MT QE-Combine under TDN queries) generated negative impact on retrieval effectiveness. In the case of MT QE-PreTrans under TDN queries, the negative effect was statistically significant. Therefore, it should be careful to use QE for MT based query translation, particularly when the queries are not short.

When comparing between the corresponding MT-based and MRD-based CLIR runs with or without PRF (see Table 4), MT-based CLIR all improved between 16%-33%. Statistical testing shows that all these improvements are statistically significant with two tailed paired-samples t-tests.

Table 4: Improvement of MT-based CLIR over MRD-based CLIR (* indicates the improvement is statistically significant)

	Baseline	PreTrans	PostTranss	Combine
T	33.27%*	32.53%*	28.31%*	26.93%*
TD	30.23%*	24.35%*	18.72%*	16.50%*
TDN	31.25%*	24.64%*	21.42%*	16.07%*

3.3 Error Analysis

When examining individual topics, we noticed that the quality of MT results were quite accurate, even for short T queries. There were many named entities in the topic statements, and Google Translate system handled them well. There were still a few named entities that Google Translate system cannot handle. But they were also out-of-vocabulary terms for the MRD. So they did not make obvious difference between MT runs and MRD runs.

However, there were 10 to 12 topics that MRD runs obtained better results than the MT runs, even though the MT translation of the queries looks better than the corresponding MRD translated queries. This probably indicates that MT generated one-best translation, although in reasonable quality, still can miss certain relevant documents which may be captured by the n-best translation approach used in MRD based CLIR. Therefore, maybe a better strategy is to combine these two approaches?

4. CONCLUSION

With further development of modern commercial statistical MT systems, it is possible to use an out-of-box commercial MT system for translating queries in CLIR task. Because many such MT systems are available on the Web, these systems can easily be used by layman Web users to build up CLIR capabilities. In this paper, therefore, we tried to examine the performance of using MT for query translation. The experiment results show that MT is a better tool for translating queries in CLIR than using MRD. Its performance is significantly better than that of using a MRD, not matter whether relevance feedback (RF) is applied or not.

When there is no help from RF, MT based query translation works the best with long queries. Its performance can reach to 99% of that of monolingual search. This is the state of the art of CLIR performance. This result may help to remove the uneasy about the worry that MT does not work under the situation that there is not much context available for translation.

However, when RF is integrated into CLIR, MT method works the best with short queries. Not only it outperforms the monolingual search under short queries, but also it achieved significant improvement over the monolingual run with both post translation based QE and combined QE methods.

Similar to that in MRD-based CLIR, when RF is helpful in MT based query translation in CLIR, it is often the post-translation QE and/or the combined QE that are the best method among the three CLIR-QE methods. In fact, post-translation QE seems to be helpful whatever the query length is, whereas pre-translation QE only works for short queries and the combined QE does not work for long queries.

Therefore, MT is not just a good query translation method in CLIR, and it actually is a better tool for query translation than MRD, not matter whether the query length are short or long, and whether RF is used or not.

Besides the main findings about using MT for query translation, our results also confirms that a specially handing of NE translation is important in CLIR, especially MRD based query translation method.

Our future work include repeat the experiments in other test collections to examine whether our findings are repeatable, and study more complex strategy of applying MT based query translation methods.

5. REFERENCES

[1] Aljlayl, M. and Frieder, O. Effective Arabic-English cross-language information retrieval via machine-readable dictionaries and machine translation. In *Proceedings of the tenth international conference on Information and knowledge management.* 2001.

[2] Artiles, J., Gonzalo, J. and Sekine, S. The SemEval-2007 WePS Evaluation: Establishing a benchmark for the Web People Search Task. In *the 4th International Workshop on Semantic Evaluations (Semeval-2007).* 2007.

[3] Ballesteros, L. and Croft, B. Dictionary Methods for Cross-Lingual Information Retrieval. *Proceedings of the 7th International DEXA Conference on Database and Expert Systems.* pages 791-801.1996

[4] Ballesteros, L. and Croft, W. B. Resolving Ambiguity for Cross-language Retrieval. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* pages 64-71.1998

[5] Bikel, D. M., Miller, S., Schwartz, R. and Weischedel, R. Nymble: a high-performance Learning Name-finder. In *Fifth Conference on Applied Natural Language Processing.* 1997.

[6] Darwish, K. and Oard, D. W. Probabilistic Structured Query Methods. In *Proceedings of the 26th International ACM SIGIR Conference on Research and Development in Information Retrieval.* pages 338-344. 2003.

[7] Grishman, R., Westbrook, D. and Meyers, A. NYU's English ACE 2005 System Description. In *ACE 2005 Evaluation Workshop.* 2005.

[8] Ji, H., Blume, M., Freitag, D., Grishman, R., Khadivi, S. and Zens, R. NYU-Fair Isaac-RWTH Chinese to English Entity Translation 07 System. In *NIST ET 2007 PI/Evaluation Workshop.* 2007.

[9] Lavrenko, V., Choquette, M. and Croft, W. B. Cross-Lingual Relevance Models. *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* page 175-182.2002

[10] Oard, D. W. A Comparative Study of Query and Document Translation for Cross-Language Information Retrieval. In *the Third Conference of the Association for Machine Translation in the Americas (AMTA).* 1998.

[11] Och, F. J. Minimum Error Rate Training for Statistical Machine Translation. In *the Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL2003).* 2003.

[12] Thomas Mandl and Womser-Hacker, C. The Effect of Named Entities on Effectiveness in Cross-Language Information Retrieval Evaluation. In *ACM SAC'05.* pages 1059-1064. 2005.

[13] Wang, J. and Oard, D. W. Combining Bidirectional Translation and Synonymy for Cross-language Information Retrieval. In *Proceedings of the ACM SIGIR 2006.* pages 202-209. 2006.

[14] Xu, J. and Weischedel, R. A Probabilistic Approach to Term Translation for Cross-Lingual Retrieval;. *Language Modeling for Information Retrieval.* W. B. Croft and J. Lafferty (Eds). 2003.

Experiments with English-Persian Text Retrieval

Abolfazl AleAhmad, Hadi Amiri, Masoud Rahgozar
Database Research Group
School of Electrical and Computer Engineering
Campus #2, University of Tehran, North Kargar St., Tehran, Iran
+982188338507
{a.aleahmad,h.amiri}@ece.ut.ac.ir, rahgozar@ut.ac.ir

Farhad Oroumchian
Department of Computer Science
University of Wollongong in Dubai
Knowledge Village, Dubai, U.A.E
+97143672400
oroumchian@acm.org

ABSTRACT
As the number of non-English documents is increasing dramatically on the web nowadays, the study and design of information retrieval systems for these languages is very important. The Persian language is the official language of Iran, Afghanistan and Tajikistan and is also spoken in some other countries in the Middle East, so there are significant amount of Persian documents available on the web. In this study, we will present and compare our English-Persian cross language text retrieval experiments on Hamshahri text collection. Also, we will present Combinatorial Translation Probability (CTP) calculation method for query translation that estimates translation probabilities based on the collection itself.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Information filtering, Query formulation, Retrieval models, Search process.

General Terms
Algorithms, Performance, Experimentation, Languages.

Keywords
Persian English cross language, Farsi bilingual text retrieval.

1. INTRODUCTION
The Persian language (also know as Farsi) is the official language of Iran, Tajikistan and Afghanistan and also second language of some other countries in the Middle East. As a result of the special and different nature of the Persian language compared to other languages like English and even Arabic, the design of information retrieval systems in Persian requires special considerations. But in spite of special characteristics of the Persian language, little efforts have been focused on retrieval of Persian text compared to other languages.

The difference between the Persian text retrieval and retrieval of other languages on the web is not just for linguistic characteristics of the languages but also because of the culture and behavior of the Persian users on the web. As an example, it is pertinent to mention the recall problem with which modern search engines are faced while queries are formulated by a Persian user. Persian uses Arabic-like script for writing and consists of 32 characters that are written mostly continuously from right to left. The problem is about two specific characters that are Kaf and Ya, each of which has four different shapes, depending on their position in words that are shown in table 1.

Table 1. Kaf and Ya Arabic words

Name	Position in Words			
	Final	Medial	Initial	Isolated
Kaf کاف	ـک or ـك	ـکـ	کـ	ک or ك
Ya یا	ـي or ـی	ـيـ	يـ	ي or ی

Kaf has two forms that are ك and ک with Unicode 1603 and 1705, also Ya has two forms that are ي (with two dots underneath) and ی (without dots) with Unicode 1610 and 1740 respectively. There is a consensus among Arab writers on the web about just using the first form of the two characters. But Persian authors of the web documents use the two forms of the characters equivalently. In order to clarify the problem, we tried a number of queries on two different search engines on 20 July 2008 and the results are summarized in table 2.

Table 2. Queries tried with different search engines

Query No.	Query Title	Search Engine	No. of documents found
1	الحليب (using ي)	Google	1,580,000
		MSN	199,000
2	الحليب (using ی)	Google	431
		MSN	10
3	شير (using ي)	Google	8,710,000
		MSN	259,000
4	شیر (using ی)	Google	1,660,000
		MSN	40,200

In table 2, queries number 1 and 2 are Arabic translation of the word 'Milk' that we ran as a sample query on the search engines. Also, queries number 3 and 4 are Persian translation of the same word. But queries 1 and 3 are written with the first form of the Ya character. One can conclude from table 2 that although queries 3 and 4 are the same from the users' perspective but the search engines return completely different set of results, however this problem can be solved by a simple preprocessing step. We also tried the queries on Yahoo and it returned consistent results that show it processes Persian queries well.

In this paper we will present our method and results of English-Persian text retrieval on Hamshahri collection. This collection is the largest Persian collection built so far which have been used by many researchers. In addition, as there is no parallel or comparable English-Persian corpora available for research, we will present combinatorial translation probability calculation method that uses terms cooccurrence in the collection for calculation of translation probabilities. Remaining parts of this paper are organized as follows: section 2 is an overview of other works on the Persian language, Section 3 describes our method for English-Persian text retrieval and the experimental results are discussed in section 4. Finally we conclude our paper in section 5.

2. RELATED WORKS ON PERSIAN

In this section we will review other researches that we have found in the literature. The authors in [10] proposed the design and testing of a Fuzzy retrieval system for Persian (FuFaIR) with support of Fuzzy quantifiers in its query language and their experiments showed that the retrieval system outperforms the vector space model. Also experiments in [2] on Hamshahri collection [7] suggest the usefulness of language modeling techniques for Persian. Furthermore in [1] the authors evaluated vector space model on Persian text with different weighting schemes and show that N-gram vector space model using Lnu-ltu weighting with slope 0.25 produces good results.

Furthermore, Compression of Persian text for web was assessed on Hamshahri collection in [12] in which the authors could obtain up to 52% reduction in size. Persian text classification is investigated on the collection in [3] and in [8] the authors have applied a reranking method, called local cluster analysis, after an initial retrieval step on the collection that shows considerable improvements on precision of the retrieval system.

3. ENGLISH-PERSIAN TEXT RETRIEVAL

There exists some monolingual Persian text retrieval studies in the literature [1, 2, 10, 8] but we could find no cross language text retrieval research on Persian. In this section we present our method for cross language text retrieval on Persian texts. More precisely, we mean the retrieval of Persian documents based on queries formulated by a human using the English language. There exist three different approaches for bilingual text retrieval according to Oard et al [11]:

1- Thesaurus-based approaches
2- Corpus-based approaches
3- Modular use of machine translation

Our method goes into the first category because we use a bilingual dictionary for translation of the query terms. Let M be the number of query terms, then we define users query as:

$$Q = \{q_i\} \ (i = 1, \ldots, M)$$

As the users query is expressed in English and the collection's documents are in Persian, we use an English-Persian dictionary with 50,000+ entries that we have prepared in our lab for translation of the query terms. Also we use a light stemmer to help matching of the query terms with the dictionary entries. If we define *T* as the translation function that returns Persian translations set of a given English term q_i, then we have $|T(q_1)| \times |T(q_2)| \times \cdots \times |T(q_M)|$ different translations for the query *Q* and as one can expect $|T(q_i)|>1$ for most of query terms. So, we need a retrieval model which enables us to take translation probabilities into consideration. This model is introduced in section 3.1 and in section 3.2 we propose our method for translation probability calculation. Also, our experimental results are presented in section 4.

3.1 Probabilistic Structured Query Method

Information retrieval systems rely on two basic statistics: the number of occurrences of a term in a document (Term Frequency or TF) and the number of documents in which a term appears (Document Frequency or DF). In case of bilingual text retrieval, when no translation probabilities are known, Pirkola's "structured queries" have been repeatedly shown to be among the most effective known approaches when several plausible translations are known for some query terms [13]. The basic idea behind Pirkola's method is to treat multiple translation alternatives as if they were all instances of the query term. Darwish and Oard later extended the model to handle the case in which translation probabilities are available by weighting the TF and DF computations, an approach they called probabilistic structured queries (PSQ) [6]. They found that Pirkola's structured queries yielded declining retrieval effectiveness with increasing numbers of translation alternatives, but that the incorporation of translation probabilities in PSQ tended to mitigate that effect. In our bilingual text retrieval experiments we use the PSQ method [6] in which TF and DF are calculated as follows:

$$\begin{aligned} TF(e, D_k) &= \sum_{f_i} p(f_i \mid e) \times TF(f_i, D_k) \\ DF(e) &= \sum_{f_i} p(f_i \mid e) \times DF(f_i) \end{aligned} \quad \textbf{(9)}$$

Where $p(f_i|e)$ is the estimated probability that *e* would be properly translated to f_i. Our method for calculation of the translation probability is presented in the next section.

3.2 Combinatorial Translation Probability

Translation probability is generally estimated from parallel corpus statistics. But as no parallel corpus is available for Persian, in this section we introduce a method which estimates English to Persian translation probabilities by use of the Persian collection itself. As most user queries contain more than two terms (e.g. in Hamshahri collection all queries has two or more terms), the main idea is to use co-occurrence probability of terms in the collection for translation probability calculation of adjacent query terms.

Consider *M* as the number of user's query terms then we define the users query as $Q = \{q_i\}$ $(i=1,\ldots,M)$. For translation of *Q*, we look up *Q* members in an English to Persian dictionary to find their Persian equivalents. Considering *T* as the translation function, then we define set of translations of *Q* members as:

$$E = \{T(q_1), T(q_2), \ldots, T(q_M)\}$$

Then the probability that two adjacent query terms q_i and q_{i+1} are translated into $E[i,x]$ and $E[i+1,y]$ respectively, is calculated from the following equation:

$$P(q_i \rightarrow E[i,x] \wedge q_{i+1} \rightarrow E[i+1,y]) = \frac{|D_{qi} \bigcap D_{qi+1}|}{c + Min(|D_{qi}|,|D_{qi+1}|)} \quad \textbf{(10)}$$

$$(x = 1..|T(q_i)|, y = 1..|T(q_{i+1})|)$$

Where D_{qi} is a subset of collection's documents, D that contains the term q_i and the constant c is a small value to prevent the denominator to become zero. In the next step we create translation probability matrix W_k for each pair of adjacent query terms:

$$W_k = \{w_{m,n}\} \ \ (m = 1..|T(q_k)|, n = 1..|T(q_{k+1})|)$$

Where $w_{m,n}$ is calculated using equation (10). Then Combinatorial Translation Probability (CTP) is a $|T(q_1)| \times |T(q_M)|$ matrix that is calculated by multiplication of all of the W_k matrices:

$$CTP(Q) = W_1 \times ... \times W_{k-1} \ \ (k = 1..M)$$

In other words, CTP matrix contains translation probability of Q members into its different possible translations in Persian. Then given $CTP(Q) = W_1 \times ... \times W_{k-1} \ (k = 1..M)$, the algorithm in table 3 returns the TDimes matrix which contains dimensions of $E = \{T(q_1), T(q_2), ..., T(q_M)\}$ matrix that correspond to top n translations of the query $Q = \{q_i\}$ $(i=1,...,M)$.

Having TDimes matrix, we are able to extract different translation of the users query from $E = \{T(q_1), T(q_2), ..., T(q_M)\}$ and their weight from CTP. For example if we consider an English query that has three terms then the most probable Persian translation of the query terms would be *E[1,TDimes [1,1]]*, *E[2,TDimes [1,2]]* and *E[3,TDimes [1,3]]* respectively and the translated query's weight would be *CTP[TopColumns[1],TopRows[1]]*.

Table 3. Calculation of the TDimes matrix

```
1.  Let TopRows[n] be the row number of n largest members
    of CTP
2.  Let TopColumns[n] be the column number of n largest
    members of CTP
3.  For i ← [1,…,n]
    3.1.  Let R = TopRows [i]
    3.2.  Let C = TopColumns [i]
    3.3.  TDimes[i,M] = C
    3.4.  For j ← [M-1,…,1]
              If (j=1)
                  Let TDimes[i,j] = R
              else
                  Let TDimes [i,j] = the culomn number of
                  the largest element of Rth row of W_{i-1}
4.  Output the TDimes matrix
```

4. EXPERIMENTAL RESULTS

4.1 Test Collection

Experimentations of this paper are accomplished by use of Hamshahri collection [7] that is now included in cross language evaluation forum (CLEF) [5] collections and to the best of our knowledge it is the largest Persian test collection. Hamshahri collection contains more that 160,000 documents which are actually news articles of the Hamshahri newspaper from year 1996 to 2002. Table 4 depicts some attributes of the collection. It should be noted that this collection have already been preprocessed in order to overcome the recall problem that we have discussed in section 1.

Table 4. Attributes of Hamshahri collection

Attributes	Value
Collection size (Unicode)	564 MB
No. Of documents	166,774
No. Of unique terms	417,339
Average length of documents	380 Terms
No. Of topics in the third topic set	50

In our experiments we use the third topics set of the collection which are prepared based on TREC specifications. TREC uses a technique called pooling [4] in which a pool of subset of documents is created for each topic and is judged for relevance by the topic author. The pooling technique prevents the relevance judgment process from biasing toward a retrieval system [14]. The topic set was also used as training set in cross language evaluation forum 2008 and contains 50 queries in both English and Persian and their relevance judgments. Each topic consists of tree parts: Title, Description and Narrative. Title contains up to two or three words which give the main idea of the topic. Description contains a full sentence or question describing the topic in short. Narrative contains a broader description of the topic including examples and perhaps mentions aspects that should not be counted as relevant.

4.2 Bilingual Text Retrieval Results

This section presents our bilingual English-Persian information retrieval results. In our experiments we just use the title of the topics. In addition, we used the Lemur toolkit for implementation of our algorithm (http://www.lemurproject.org/). The default retrieval model of the lemur's retrieval engine (Indri) is the language modeling.

Our bilingual experimental results include three different runs that are summarized in table 5. Figure 1 depicts the precision-recall graph of their results that are calculated by use of the Treceval tool [9]. In the first run we retrieve documents based on Persian title of the 50 topics as a base line to compare other results with it. Our second run uses the English version of the 50 topics' titles and we translate them by use of the English-Persian dictionary. Then we formulate a query in Persian by concatenation of all meanings of each of the query terms.

The Third run uses the retrieval model of section 3.1 and calculation of translation probabilities with method of section 3.2. In other words, we use the English version of the 50 topics' titles and apply the method of section 3.2 for translation of the topics and calculation of translation probabilities. Then we formulate Persian equivalents of the 50 English queries based on top 5 translations with highest probability.

The Indri retrieval engine supports structured queries, so we could easily implement the PSQ method using CPT for weighting.

Table 5. Runs and their description

Run#	Run Name	Description
1	Monolingual	Monolingual LM retrieval model
2	All Meanings	Bilingual English-Persian LM retrieval model using all meanings of each of the query terms
3	PSQ CPT Top5	Bilingual English-Persian LM retrieval model using methods of 3.1, 3.2 sections

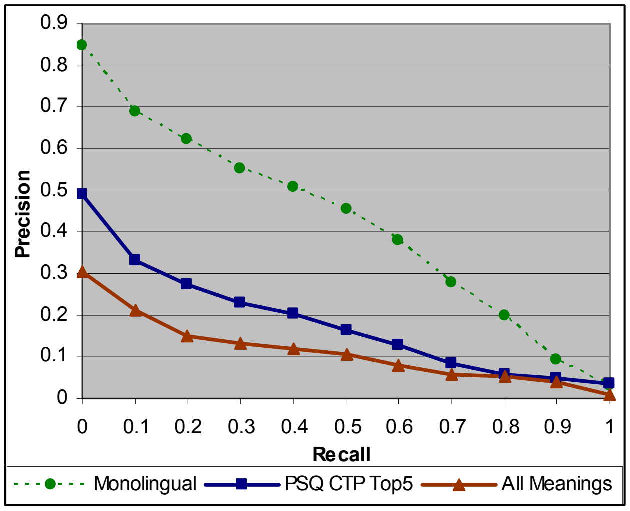


Figure 1. Precision-Recall of the three runs

Comparison of the three runs shows that performance of the PSQ CTP Top5 run is considerably better that the All Meanings run and our proposed method can work well in case of lack of language resources like parallel corpora. Also, performance of PSQ CTP Top5 is comparable with performance of other cross language text retrieval results that other researchers have reported on other languages [6].

5. CONCLUSION AND FUTURE WORKS

In this paper we proposed a method for calculation of translation probabilities based on statistics of the collection itself. The results that we obtained form this method shows that it is able to improve cross language retrieval performance in cases that we do not have enough language resources for calculation of translation probabilities.

Hamshahri collection is now standardized according to CLEF standards and 50 new bilingual queries are developed for the collection recently that facilitates study of bilingual English-Persian text retrieval. So, one of our future work would be investigation of other aspects of cross language information retrieval on the Persian language.

6. ACKNOWLEDGEMENT

This research was partially supported by Iranian Telecommunication Research Center (ITRC). Also, we would like to thank members of DBRG lab at University of Tehran that helped us in this research.

7. REFERENCES

[1] A. Aleahmad, P. Hakimian, F. Mahdikhani, F. Oroumchian. N-Gram and Local Context Analysis for Persian Text Retrieval. International Symposium on Signal Processing and its Applications ISSPA 2007, Sharjah, UAE, 2007.

[2] H. Amiri, A. AleAhmad, F. Oroumchian, C. Lucas, M. Rahgozar, Using OWA Fuzzy Operator to Merge Retrieval System Results. In Computational Approaches to Arabic Script-based Languages (CAASL 2007), Stanford University, USA, 2007.

[3] B. Bina, M. Rahgozar, A. Dehmoubed. Automatic Classification of Persian Text. In 13th International CSI Computer Conference (CSICC) 2008, Kish island, Iran, 2008.

[4] N. Craswell, D. Hawking, R. Wilkinson, and M.Wu. Overview of the TREC 2004 web track. In Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004), 2004.

[5] Cross Language Evaluation Forum (CLEF). http://www.clef-campaign.org/

[6] Kareem Darwish and Douglas W. Oard. Probabilistic structured query methods. In Proceedings of the 21st Annual 26th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 338–344. ACM Press, July 2003.

[7] Hamshahri Collection. http://ece.ut.ac.ir/dbrg/hamshahri/

[8] A.H. JadidiNezhad, H. Amiri. Local Cluster Analysis as a Basis for High-Precision Information Retrieval. The 6th International Conference on Informatics and Systems, INFOS 2008, Cairo, Egypt, 2008.

[9] National Institution of Standards and Technology, http://trec.nist.gov/trec_eval/

[10] A. Nayyeri, F. Oroumchian. FuFaIR: a Fuzzy Persian Information Retrieval System. IEEE International Conference on Computer Systems and Applications, pp. 1126-1130, 2006.

[11] D. W. Oard & B. J. Dorr, A Survey of Multilingual Text Retrieval, UMIACS TR-96-19, University of Maryland, College Park, MD, 1996.

[12] F. Oroumchian, E. Darrudi. Experiments with Persian Text Compression for Web. World Wide Web 2004, pp. 478-479, New York, New York, USA, May 2004.

[13] Ari Pirkola. The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 55–63. ACM Press, August 1998.

[14] J. Zobel. How reliable are the results of large-scale information retrieval experiments? In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 307–314, Melbourne, Australia. ACM Press, New York, August 1998.

How do Greeks Search the Web? A Query Log Analysis Study.

Efthimis N. Efthimiadis
The Information School
University of Washington
Seattle, WA, USA
efthimis@u.washington.edu

ABSTRACT

This study investigates how Greeks search the web through the analysis of search logs from the AltaVista search engine. The analysis looks at the data as a whole and also examines the use of the Greek language in web searching. Statistics include frequencies of queries, term distributions, languages used including mixed language queries, search features used in query formulation and query length patterns.

Categories and Subject Descriptors

H3.3. Information storage and retrieval: Information search and retrieval; *Query formulation*; *Search process*
H.3.5 Online Information Services
H5.4: Hypertext/Hypermedia. – User issues.

General Terms

Measurement, Experimentation, Human Factors.

Keywords

Greek web search, query log analysis, user behavior, non-English web search.

1. INTRODUCTION

The research presented in this paper is a characterization study of how Greeks search the web by examining a set of web search logs. The analysis is based on a sample of two months of web search logs from the AltaVista search engine. The analysis looks at the data as a whole and examines the use of the Greek language in searching. Statistics include frequencies of queries, term distributions, languages used including mixed language queries, search features used in query formulation and query length patterns.

Questions addressed include:

- How often was AltaVista used during the study period?
- What queries were the most popular?
- How many queries were there?
- How many terms per query?
- What languages were used?
- How complex were the queries?
- Did users employ advanced search capabilities?

2. RELATED WORK

Studying web search behavior through characterization studies of search by analyzing search logs is well established [1]. Large scale studies based on AltaVista [2] and AOL [3] logs provide some basis for comparisons, though the data is primarily US-focused. Jansen and Spink provide a comparison of nine web search studies from five search engines [4]. Anick investigated how people develop search formulations and utilize AltaVista's Prisma feature that supported query modification and query expansion [5]. Baeza-Yates *et al.* discuss how query log mining can facilitate the discovery of user behavior patterns and assists search engines in serving better results [6]. Teevan *et al.* studied repeat queries in search logs and established its importance for users [7].

3. METHODOLOGY

The study dataset is a sample of web search log data from searches that originated from Greece. The searches were done using the AltaVista search engine. The dataset consists of about 2.5 million queries from January 1 to February 28, 2004. The information that is provided in the logs includes the following fields:

anonymized "user" identifier, date, time, query
U1234, 02/01/04, 13:52:22.258, physics

The logs were text-processed and then statistically analyzed using SPSS and R.

4. RESULTS and DISCUSSION

In this paper the results are presented with a minimal comparison to other studies. Any such comparisons are primarily targeted in relating the specialized results from the Greek web to the more general studies.

This dataset and the results derived from this study are valuable because there are not query logs available specifically for Greece. Further, the search log datasets that are available, such as, Excite, AOL, MSN Live, are primarily from the United States, and the research community is painfully aware of the difficulties in accessing search logs due to user privacy concerns.

4.1 Users

The number of unique users who posed queries in January-February 2004 was 148,058. There were 84,261 users in January and 79553 users in February. Table 1 shows the top ten ranked number of searches per user over the two months. The user that conducted the largest number of queries searched 3,745 queries during the study period. There were 18,253 users who searched only once, which represents 12.33% of the unique users. One query was searched by as many as 2,341 (1.58%) users. The average number of queries searched per user is 17.27 (sd=43.86).

Table 1: Top ten users with highest frequency of queries searched.

Rank	Number of Queries
1	3745
2	2383
3	2321
4	2105
5	1981
6	1751
7	1631
8	1594
9	1394
10	1293

Figure 1 shows the frequency rank distribution of users and their queries. This graph uses a random sampling of 10% of the data points in the data set. The most prolific user searched just over 60 searches a day. It is very likely that these top ranked users of Table 1, are not individuals, but shared workstations in public places, such as universities or internet cafés.

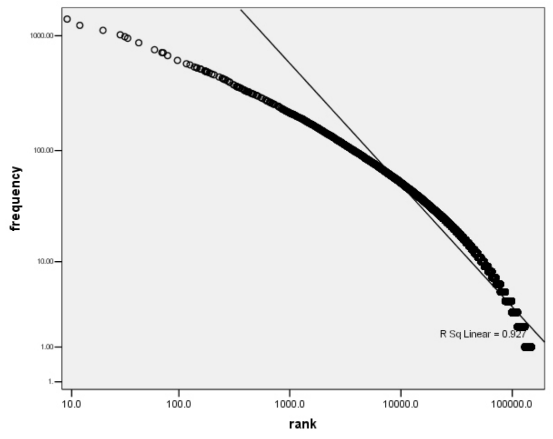


Figure 1: Frequency rank distribution of queries

4.2 Query Characteristics

Basic statistics of the entire dataset, that include both Greek and foreign language queries, are given in Table 2. The total number of queries is 2,555,364. The number of single word queries is 620,429 (24.28%). The number of unique queries is 619,461 (24.24%). The number of queries that occurred only once is 511,410 (20.01%). The average query frequency is 4.13. The total number of words in all queries is 6,910,796. The total number of unique words is 337,704. The total number of unique words after case normalization is 157,519.

The total number of queries containing Greek is 167,669 (6.56% of the total number of queries). The number of single word Greek queries is 79,132 (47.20%). The number of unique Greek queries is 48,098 (28.69%). The number of Greek queries that occurred once is 39,110 (23.33%). The average query frequency is 3.49. The total number of Greek words used is 304,949. The total number of unique Greek words used is 43,648.

The number of unique users who searched using Greek in their queries is 17,828 (12.05% of the 148,058 users).

Table 2: Query characteristics

	All Queries	Greek Queries
number of queries	2,555,364	167,669 (6.56%)
single word queries	620,429 (24.28%)	79,132 (47.20%)
unique queries	619,461 (24.24%)	48,098 (28.69%)
queries occurring once	511,410 (20.01%)	39,110 (23.33%)
average query frequency	4.13	3.49
total number of words	6,910,796	304,949
unique words	337,704	43,648

4.3 What do Greeks Search For?

The queries were ranked by frequency of occurrence for the two month period. The top 20 queries are presented in Table 3. The first two columns of the table show the most frequent queries including queries with adult content, whereas column three and four present the top queries excluding adult content. The top 20 queries are all in English, and the most frequent Greek query occurred in Rank 22.

Table 3: Top 20 most frequent queries

Frequency	Query (incl. adult content)	Frequency	Query (sanitized)
8748	sex	1287	lyrics
2356	porno	1093	cracks
1745	sex free sex	1036	kazaa lite
1287	lyrics	1001	orlando bloom
1093	cracks	870	greece
1036	kazaa lite	866	kazaa
1001	orlando bloom	858	crack
943	sex free sex pics	826	Games
912	porn	784	In.gr
870	greece	749	wallpapers
866	kazaa	734	google
858	crack	718	woodcarving
847	sex free porn	694	cheats
826	games	598	lingerie
821	free sex	593	www.in.gr
784	in.gr	558	infinite
749	wallpapers	502	greek celebrities
734	google	498	free sms
718	woodcarving	487	Cd covers
694	cheats	456	lord of the rings

It is evident from the above table the top ranked queries are for porn and music. Kazaa and music sharing and downloading was big in 2004 all around the world. Notice that Google and in.gr, a major Greek portal, are searched for as navigational queries on the AltaVista search engine. These findings are consistent with the findings of other studies, where sex and entertainment were reported as dominant [8].

The Greek language queries account for 6.56% of the total number of queries in the sample. Table 4 lists the 20 most frequent Greek queries. Similar to the English queries the content of the queries is primarily relating to sex, entertainment, and cooking. Since the

Olympic Games were in Athens in 2004 this is another highly ranked query.

By studying the Greek queries one observes that semantically same queries are repeated using accents, diacritics, or upper/lower case, e.g., ζωδια and ΖΩΔΙΑ. As it has been established in previous research such queries produce different results as the search engines do not take into account Greek language specifics [9, 10]

Table 4: Top 20 most frequent Greek queries

Freq	Query
350	στοιχημα
285	ζωδια
280	Νινο
279	σεξ
238	ηλεκτρονικό εμπόριο
230	νεα
215	ΣΥΝΤΑΓΕΣ
207	συνταγές μαγειρικής
191	μ***ι
179	ΖΩΔΙΑ
171	φωτιστικά
169	ΣΕΞ
161	μ***ί
157	πορνο
141	ΟΛΥΜΠΙΑΚΟΙ ΑΓΩΝΕΣ
139	ΠΕΡΙΒΑΛΛΟΝ
137	Αρτα
135	χανια
130	περιοδικα
128	ΑΝΕΚΔΟΤΑ

4.4 Query Formulation Characteristics

The queries were analyzed in order to establish the patterns of advanced search query formulation and syntax used by the searchers. The AltaVista advanced search operators and query syntax was used in the analysis.

The total number of queries containing logical operators is 5,275 (0.21%). Examples of the operators used are: "AND", "OR", "AND NOT", "NEAR" as shown in the example queries: "eric hobsbawn" AND bibliography; hcv AND genotype AND sequence; Hotel NEAR Spagna NEAR Rome; cars AND NOT fiat.

The total number of queries containing other advanced search operators is 17,594 (0.69%). Examples of these operators are: “anchor:”, ”applet:”, “object:”, “domain:”, “host:”, “image:”, “like:”, “link:”, “text:”, “title:”, “url:”, etc. Example queries were: like:http://www.biblionet.gr/; link:www.ert.gr.

The total number of queries containing double quotes, which indicates phrase searching, is 379,089 (14.84%). For example, "honda civic coupe". The total number of queries starting with double quote (“…”) is 85,448, for example, "University of Portsmouth"; "uk university".

Users often made syntax errors in their attempts to use advance operators. For example, AltaVista requires that operators are typed in uppercase, so when typed in lower case are treated as search terms or stop words. The total number of queries containing wrong operators is 122,608 (e.g., lower case 'and', 'not', 'or', 'and not', 'near', ‘*’, ‘.’). Examples of errors are:

organic solar cells and polymers
+Macedonia and +(state or government)
free not trial games
breast near enlargement near exercise and not "breastsuccess"
"escrow account" and not mortgage
greek or music
greek near music
*.pdf

4.5 Query Length

How many words are included in each query? Is there a difference between the way Greeks construct English or Greek language queries? Figure 2 shows that on average the query length of both English and Greek queries is 1.8 terms per query.

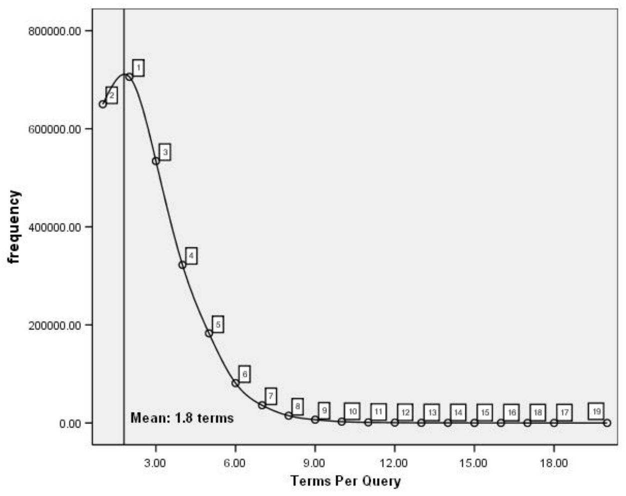


Figure 2: Query length distribution

The analysis of all queries shows that the query length ranged from a minimum of 1 to a maximum of 49, with a mean of 1.87 and standard deviation of 1.65. Most queries are in the range of 1 to 6 terms. A handful of queries were identified that had more than 400 terms long. In cases such as this the person had taken entire paragraphs of text and searched for them. These queries were excluded from the analysis as being outliers that would have skewed the results. Table 5 gives the breakdown of the number of queries per query length.

Table 5: Number of queries per query length

rank	# of terms	frequency
1	2	705812
2	1	650067
3	3	534170
4	4	322415
5	5	182860
6	6	81330
7	7	36150
8	8	14775
9	9	6664

The analysis of the Greek only queries shows that the query length ranged from a minimum of 1 to a maximum of 25, with a mean of

2.12 and standard deviation of 1.2. Figure 3 shows the query length distribution of the Greek only queries. It can be seen that there is a similar pattern in the distribution of query length as with the whole dataset. In both cases two word queries are the majority followed by one word queries and then there is a significant drop from 3 to 6 word queries, and gradually leveling off after that.

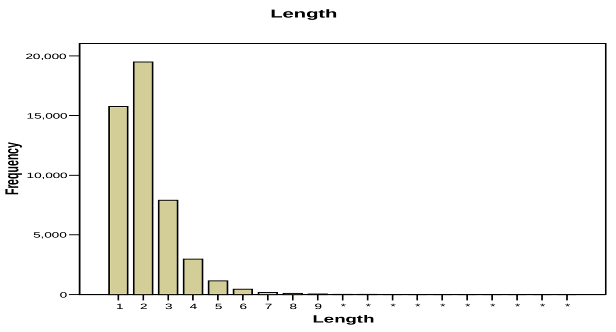


Figure 3: Query length distribution of Greek queries

4.6 What Languages do Greeks Search in?

A random sample representing one third of the dataset was used to establish the languages used in the queries. The distribution shows that:

90% of Greeks search using a Latin–based character set
8% of Greeks search using the Greek character sets
2% of Greeks use some other encoding

English, French, German, Italian seemed to be the most used languages. This corresponds well with the fact that Greeks learn those languages, as a second language, and a large number of Greeks have studied for an academic degree in the respective countries.

In the 2% of other encodings the number of languages identified included Cyrillic-based languages, such as Bulgarian, Russian, and Serbian, as well as Arabic, and Hebrew (reflecting the sizable immigrant groups from these countries).

5. CONCLUSIONS

This study provides preliminary results of the search behavior of Greeks as found in a sample search log from AltaVista. There were 148,058 users who searched over 2.5 million queries. The percentage of Greek queries was a very low 6.56%. The average query length of all queries was 1.8 terms, while the average length for Greek queries was 2.12 terms. It is believed that the number of query terms will increase as Internet penetration and web search becomes more widely accepted throughout Greece. These results are consistent with findings of earlier web search log studies [1].

The analysis of the queries revealed that a very small number of queries had any advanced search features. Phrase searching was the most employed feature with 14.84%. Logical operators were successfully employed in 0.21% of the queries, and advance operators about 0.69%. Syntax errors had a relatively higher rate, 4.8%. Such failures are cause of great frustration for users who get dissatisfied with the inability of search engines to correctly interpret their intent.

The finding that Greeks search primarily using Latin-based languages rather than Greek is not surprising. The reasons for this could be the inadequate treatment of the Greek language as well as the level of indexing of the Greek Web by search engines [9, 11]. Efthimiadis *et al.* [9, 11] showed that about one in three of the Greek navigational queries fail. As non-English web search proliferates the hope is that search engines will take these language issues into consideration.

Further analysis of the data will report on session segmentation, query reformulation, and query overlap.

Acknowledgements
I would like to thank S. Hsueh and S. Hendriks for their assistance at different stages of this research project.

6. REFERENCES

1. Spink, A. and B.J. Jansen, *Web Search: Public Searching of the Web*. Information Science and Knowledge Management Vol. 6. 2004: Springer.
2. Silverstein, C., et al., *Analysis of a very large Web search engine query log.* SIGIR Forum, 1999. **33**(1): p. 6-12.
3. Pass, G., A. Chowdhury, and C. Torgeson, *A picture of search*, in *Proceedings of the 1st international conference on Scalable information systems*. 2006, ACM: Hong Kong.
4. Jansen, B.J. and A. Spink, *How are we searching the World Wide Web? A comparison of nine search engine transaction logs.* Information Processing &amp; Management, 2006. **42**(1): p. 248-63.
5. Anick, P.G. *Using Terminological Feedback for Web Search Refinement - A Log-based Study*. in *the 26th annual international ACM SIGIR conference on Research and Development in Information retrieval*. 2003.
6. Baeza-Yates, R., et al., *Modeling User Search Behavior*, in *Proceedings of the Third Latin American Web Congress*. 2005, IEEE Computer Society.
7. Teevan, J., et al., *Information re-retrieval: repeat queries in Yahoo's logs*, in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 2007, ACM: Amsterdam, The Netherlands.
8. Spink, A., et al., *From e-sex to e-commerce: Web search changes.* Computer, 2002. **35**(3): p. 107-9.
9. Efthimiadis, E.N., et al., *An Evaluation of How Search Engines Respond to Greek Language Queries*, in *Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*. 2008, IEEE Computer Society: Hawaii. p. 136.
10. Lazarinis, F., *Web retrieval systems and the Greek language: do they have an understanding?* Journal of Information Science, 2007. **33**(5): p. 622-636.
11. Efthimiadis, E.N., et al., *Non-English Web Search: An Evaluation of Indexing and Searching the Greek Web* Information Retrieval (in press) 2009.

LearnLexTo: A Machine-Learning Based Word Segmentation for Indexing Thai Texts

Choochart Haruechaiyasak, Sarawoot Kongyoung, and Chaianun Damrongrat
Human Language Technology Laboratory (HLT)
National Electronics and Computer Technology Center (NECTEC)
Thailand Science Park, Klong Luang, Pathumthani 12120, Thailand
{choochart.haruechaiyasak, sarawoot.kongyoung, chaianun.damrongrat}@nectec.or.th

ABSTRACT

Thai language is considered as an unsegmented language in which words are written continuously without the use of word delimiters. To index Thai texts via the inverted index, a word segmentation algorithm is usually required to tokenize a text into a series of terms. Recent works on word segmentation reported Conditional Random Fields (CRFs) as the best machine learning algorithm, outperforming the dictionary-based approach and other machine learning algorithms. Our main contribution is to propose a new hybrid approach, *LearnLexTo*, which further improves the CRF model by integrating the dictionary-based approach. The key idea is to solve the ambiguity problem in the CRF model by using the dictionary-based approach which relies on a valid word set. Experimental results showed that the proposed hybrid approach yields the highest F_1 value of *88.46%*, compared to *82.07%* by using the dictionary-based approach and *85.71%* by using the CRF model.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing – *linguistic processing*

General Terms

Algorithm, Experimentation, Performance

Keywords

Indexing, tokenization, word segmentation

1. INTRODUCTION AND BACKGROUND

The process of text indexing for Latin-based languages such as English is very straightforward. Given an English text, terms can be easily tokenized by observing word delimiters such as whitespace and special characters. The tokenized terms are then stored into the *inverted index* structure which provides storage efficiency and fast retrieval [1].

In this paper, we explore some challenging issues in indexing Thai texts. Thai texts are naturally unsegmented, i.e., words are written continuously without the explicit use of word delimiters. To index Thai texts via the inverted index structure, a word segmentation algorithm is typically required to perform the term tokenization. The main task of word segmentation is to identify word boundaries on a given input text string. Although there are many previously proposed techniques for word segmentation task, none of them yielded perfect results. The reason is due to the *unknown word* and *ambiguity* problems which typically occur while parsing and segmenting the input text string [2]. Unknown words refer to words which are not included in the dictionary while performing the word segmentation process. Whereas, the ambiguity problem occurs when there are more than one way to segment a given text.

Current widely applied techniques for Thai word segmentation can be broadly classified into two types, *dictionary-based* and *machine learning*. The dictionary-based approach relies on a set of terms from a dictionary for parsing and segmenting input texts into word tokens. The dictionary-based approach is relatively simple and straightforward. During the parsing process, series of characters are looked up on the dictionary for matching terms. The performance of the dictionary-based approach depends on the quality and size of the word set in the dictionary used during the segmentation process.

To solve the unknown word problem in the dictionary-based approach, one possible solution is by using language-dependent lexical rules to merge unknown segments. This solution has the main drawback of the requirement to update the lexical rules. In modern days, many new words are generated through the transliteration of foreign words. As a result, the process of updating the lexical rules becomes inefficient. Another alternative is to collect and include as many unknown words into the dictionary. However, this solution is not dynamic and scalable since it requires manual addition of new words.

The ambiguity problem for the dictionary-based approach can be solved via several selection techniques such as by selecting the longest possible term, i.e., longest matching, and by selecting the segmented series which yields the minimum number of word tokens, i.e., maximal matching. Previous work reported that the maximal matching algorithm has a marginal improvement over the longest matching algorithm, however, with the tradeoff on longer running time [7]. Both longest matching and maximal matching algorithms can be considered as using some heuristics to solve the ambiguity problem. The main disadvantage is these heuristics are very simple and static, i.e., unable to adapt to changing domains.

Due to the drawbacks of the dictionary-based approach, machine learning algorithms have been adopted for word segmentation task. Using a tagged corpus in which word boundaries are explicitly marked with a special character, a machine learning algorithm could be applied to train a model based on features surrounding these boundaries. The main advantage of the machine learning approach is the independency of dictionary. The unknown word and ambiguity problems are handled by the classification model which is learned from various character patterns inside a tagged corpus. Therefore, the performance of this approach depends on the domain and the corpus size. For example, if a model is constructed based on a specific domain corpus, it might not perform well on other different domains. Moreover, the corpus must be large enough to cover all different character patterns so that the model could be trained effectively.

Recently the Conditional Random Fields (CRFs) algorithm has emerged as a novel approach which yields better performance than other machine learning algorithms for the task of labeling and segmenting sequence data [5, 6]. The CRF model has the main advantage of ability to learn various word patterns by conditionally assigning different weights on the feature sets. Some examples of previous works which applied the CRFs in performing word segmentation and morphological analysis are as follows. Kudo et al. applied the Conditional Random Fields (CRFs) to solve the ambiguity problem in Japanese morphological analysis [4]. The results showed that the CRF model achieved better performance than the HMMs and MEMM models. Peng et al. applied the linear-chain CRF model for Chinese word segmentation [6]. Their proposed model included a probabilistic new word detection method to further improve performance.

Kruengkrai et al. applied the CRF algorithm to train a word segmentation model for Thai language [3]. Their work mainly focused on solving the ambiguity problem in word segmentation. Two path selection schemes based on confidence estimation and Viterbi were proposed. The feature set used in their model required the POS tagging information. Therefore, if the POS tagging is inaccurate, the performance of the word segmentation is decreased. In our CRF model, we construct the feature set based on the character types of the n-gram characters surrounding the word boundary. As shown from the experiments, the character types in Thai language provide effective information for training a word segmentation model.

Our main contribution is to propose a new hybrid word segmentation approach by integrating the CRF model with the dictionary-based approach. Although the CRF model could effectively handle the unknown word problem, it sometimes generates large segments which overlap the actual word boundary. Therefore, the dictionary-based approach is used to check and correct the ambiguity case, i.e., overlapping segments, output from the CRF model. The experimental results showed that the new hybrid approach yielded the highest F_1 value compared to the dictionary-based approach and the basic CRF model.

2. THE PROPOSED METHOD

In this section, we describe our proposed word segmentation approach called *LearnLexTo*, i.e., a machine-learning based lexeme tokenizer. From our initial experiments, the CRF model yielded better performance than the dictionary-based approach. Therefore, we apply the CRF model as the main module for performing the word segmentation task. CRFs are undirected graphical model for segmenting and labeling structured data. The full details of the CRFs algorithm can be found in Lafferty et al. [5]. In this paper, we use the open-source CRF tool called *Pocket CRF*[1] to perform our experiments. The Pocket CRF applies the limited memory quasi-Newton gradient-climber method (LBFGS) for fast training. Under the CRF model, the word segmentation problem can be formulated as a binary classification task in which each character in the text string is predicted into one of two classes: the beginning of a word and intra-word characters. Using a large tagged corpus, we could train a CRF model based on the features surrounding each tag class.

To prepare the feature set for Thai word segmentation, we distinguish characters into ten different types as shown in Figure 1. The set of character types is designed based on linguistic knowledge, i.e., lexical rules, of Thai language. These character types provide effective information for identifying the word boundaries. Each data instance contains an n-gram of characters preceding and following the current character. For example, in a 3-gram data set, previous, current and next characters are considered during the model training.

Tag	Type	Value
c	Consonant characters which can be assigned as the word ending character	ก ข ฃ ค ฆ ง จ ช ซ ญ ฎ ฏ ฐ ฑ ฒ ณ ด ต ถ ท ธ น บ ป พ ฟ ภ ม ย ร ล ว ศ ษ ส ฬ อ
n	Consonant characters which **cannot** be assigned as the word ending character	ฅ ฉ ผ ฝ ฌ ห ฮ
v	Vowel characters which are **not** allowed to begin a word	ะ ิ ี ึ ื ุ ู ั า ำ ๅ
w	Vowel characters which are allowed to begin a word	เ แ โ ใ ไ
t	Tonal characters	่ ้ ๊ ๋
s	Symbol characters	์ ๆ ฯ .
d	Digit characters	0-9
q	Quote characters	'-' “-”
p	Space character within a word	_
o	Other characters	a-z A-Z

Figure 1: Feature set based on character types for constructing the CRF model

The results from the initial evaluation showed that the CRF model performs well in the case of unknown words. However, the CRF model has the tendency to generate large segments which overlap multiple words, i.e., ambiguity problem. Figure 2 illustrates three problem cases resulted from both the dictionary-based approach (denoted by DCB) and the CRF model (denoted by CRF). The word boundaries are clearly marked with the vertical bar character and incorrect segments are underlined. Case I and II show the ambiguity problem in the CRF-based approach. In Case I, the CRF model generates large segments which completely overlap two words. It can be observed that the dictionary-based (DCB) approach yields totally correct segments. In Case II, the CRF model generates segments which partially

[1]http://pocket-crf-1.sourceforge.net/

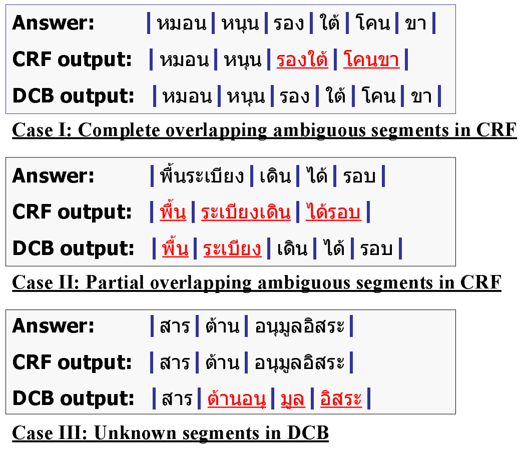

Answer: | หมอน | หนุน | รอง | ใต้ | โคน | ขา |
CRF output: | หมอน | หนุน | รองใต้ | โคนขา |
DCB output: | หมอน | หนุน | รอง | ใต้ | โคน | ขา |
Case I: Complete overlapping ambiguous segments in CRF
Answer: | พื้นระเบียง | เดิน | ได้ | รอบ |
CRF output: | พื้น | ระเบียงเดิน | ได้รอบ |
DCB output: | พื้น | ระเบียง | เดิน | ได้ | รอบ |
Case II: Partial overlapping ambiguous segments in CRF
Answer: | สาร | ต้าน | อนุมูลอิสระ |
CRF output: | สาร | ต้าน | อนุมูลอิสระ |
DCB output: | สาร | ต้านอนุ | มูล | อิสระ |
Case III: Unknown segments in DCB

Figure 2: Example of segmenting result analysis

overlap two words, whereas, the dictionary-based approach yields better segmenting results.

To solve the ambiguity problem generated from the CRF model, we apply the dictionary-based approach to check the output segments returned from the CRF model. The dictionary-based approach, which relies on a valid word set, has the tendency to generate smaller segments which are most likely matched with the correct words in the answer. Therefore, the dictionary-based approach could help improve the performane of the basic CRF model.

However, the dictionary-based approach sometimes faces the unknown segments. As previously discussed, the dictionary-based approach could not handle the unknown word problem well compared to the CRF model. Therefore, to improve the overall performance of the hybrid approach, we include the unknown segment checking during the dictionary-based process. If, during the segmentation, an unknown segment is detected, the dictionary-based process is ignored and the original results from the CRF model are used. Case III illustrates an example of unknown segments in the dictionary-based approach. By performing the unknown segment checking, the result from the dictionary-based process is ignored. The final segmenting result is returned from the CRF model which is totally correct. The overall process for the proposed hybrid approach is summarized in Figure 3.

3. EXPERIMENTS AND DISCUSSION

We used ORCHID corpus[2] to evaluate the performance among different word segmentation approaches. The ORCHID corpus is an open corpus which is widely used and well recognized among Thai NLP community. The ORCHID corpus contains *113,404* manually segmented words. We randomly split the corpus into training and test sets, each containing 80% and 20%, respectively.

Three measures of precision, recall and F_1 are used for performance evaluation. Precision is the number of tokens correctly segmented by the algorithm divided by the total number of tokens returned from the algorithm. Recall is the number of tokens correctly segmented by the algorithm divided by the total number of tokens in the test set. F_1 measure is the harmonic mean of precision and recall

[2]http://links.nectec.or.th/orchid

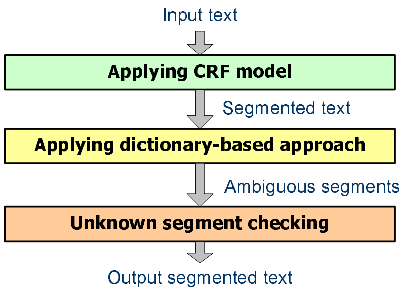


Figure 3: The overall process under the hybrid approach

We first evaluate the performance of the CRF approach. To train CRF models, we vary the number of grams on the feature sets from *5* to *13*. Table 1 presents the performance evaluation based on precision, recall and F1 measures. It can be observed that the performance of the CRF approach is improved when the number of n-grams increases. The reason is the larger number of grams could provide more evidence for training the model. However, the best performance was obtained when the number of grams is equal to *11*.

Table 1: Evaluation results of the CRF-based approach

Number of grams	Precision	Recall	F_1
5	82.47	83.62	83.04
7	84.98	85.46	85.22
9	85.31	85.92	85.62
11	**85.42**	**85.99**	**85.71**
13	85.03	85.66	85.34

For the dictionary-based approach, we used the LEX*i*TRON[3] which contains approximately *30,000* words as the main dictionary. To observe the effect of using a domain-specific word set, we compared the performance of the dictionary-based approach by using (1) general word set from LEX*i*TRON (denoted by *DCB (Lex)*), (2) by using the word set obtained from the training corpus (denoted by *DCB (Train)*), and (3) by using the combined word sets from both LEX*i*TRON and the training corpus (denoted by *DCB (Lex + Train)*).

Table 2: Evaluation results of the dictionary-based approach

Approach	Precision	Recall	F_1
DCB (Lex)	70.84	72.19	71.51
DCB (Train)	**80.21**	**84.02**	**82.07**
DCB (Lex+Train)	75.99	73.68	74.82

Table 2 shows the evaluation results of the dictionary-based approach by using different word sets. It can be observed that using the general word set from LEX*i*TRON yields the worst performance. On the other hand, using the word set from the training corpus significantly improves the performance. The reason is because the word set from the training corpus offers better word coverage related to the texts being segmented. However, the combination of

[3]http://lexitron.nectec.or.th

both word sets from the LEX*i*TRON and the training corpus yields the worse performance than using only the word set from the training corpus. This is due to the ambiguity problem since larger word set would generate more selection paths and the algorithm may select the wrong word choice.

Next, we evaluate the performance of the proposed hybrid approach. For the CRF model, we set the number of grams for the feature sets equal to *11*. As mentioned earlier, the CRF model could generate ambiguous segments which lower the overall performance of word segmentation process. The dictionary-based algorithm is then applied to solve the ambiguity problem in the CRF approach.

To observe the effect of using different word set, we compare the performance of the hybrid approach by using the CRF model and the dictionary-based approach with three different word sets: (1) the general word set from LEX*i*TRON (denoted by *HBD (Lex)*), (2) the word set obtained from the training corpus (denoted by *HBD (Train)*), and (3) the combined word sets from both LEX*i*TRON and the training corpus (denoted by *HBD (Lex+Train)*). The evaluation results of the hybrid approach is presented in Table 3. It can be observed that using the combined word sets from LEX*i*TRON and the training corpus yields the best performance. This result contradicts the previous finding that for the dictionary-based approach in which using the word set from the training corpus yielded the best performance. The reason is more word coverage offers particular advantage in solving the ambiguous segments generated by the CRF model.

Table 3: Evaluation results of the hybrid approach

Approach	Precision	Recall	F_1
HBD (Lex)	84.50	90.72	87.50
HBD (Train)	83.54	90.45	86.86
HBD (Lex+Train)	**84.68**	**90.88**	**87.67**

Next, we improve the performance of the hybrid approach further by performing the unknown segment checking for the dictionary-based approach. Under our proposed hybrid approach, the ambiguous segments from the CRF model are forwarded and handled by the dictionary-based approach. However, the dictionary-based approach could generate unknown segments from this ambiguous segment. If an unknown segment is found, we ignore the dictionary-based approach and use the segmenting results from the CRF model. We refer to this process as unknown segment checking (denoted by *HBD-UNK*).

Table 4 shows the results by incorporating the unknown segment checking for the hybrid approach. It can be observed that by including the unknown segment checking, the performance based on the F_1 measure is improved to *88.46%* as compared to *87.67%* by using the original hybrid approach. This improvement is relatively small since the ORCHID corpus does not contain many unknown words. However, we believe that the process of unknown segment checking is very useful in general since typical texts obtained from various sources, such as Web pages, usually contain many unknown words.

In summary, the dictionary-based approach yields the maximum F_1 measure equal to *82.07%*. The CRF approach gives the maximum value for F_1 measure equal to *85.71%*. The proposed hybrid approach yields the maximum value for F_1 measure equal to *87.67%*. The improved hybrid approach by incorporating the unknown segment checking yields the best performance of F_1 measure equal to *88.46%*.

Table 4: Improving hybrid approach with unknown segment checking

Approach	Precision	Recall	F_1
HBD-UNK	**86.02**	**91.05**	**88.46**

4. CONCLUSIONS

In this paper, we proposed a new hybrid word segmentation approach called *LearnLexTo* by integrating the dictionary-based approach with the CRF model. The proposed approach first uses the CRF model to perform the segmentation. The ambiguous segments from the CRF model are forwarded and handled by the dictionary-based approach. Our approach also includes the unknown segment checking for the dictionary-based approach which help improve the overall performance. Based on the evaluation, the performance of the proposed hybrid appproach based on the F_1 measure is equal to *88.46%* compared to *82.07%* by using the dictionary-based approach and *85.71%* by using the CRF model.

Our future works include a construction of Thai IR evaluation corpus similar to the TREC retrieval track corpus. The corpus would allow us to qualitatively evaluate different word segmentation algorithms on many aspects of information retrieval (IR) processes such as indexing and query processing.

5. REFERENCES

[1] W. Frakes and R. Baeza-Yates (eds.), *Information Retrieval: Data Structures and Algorithms*, Prentice Hall, 1992.

[2] C. Haruechaiyasak et al., "A Collaborative Framework for Collecting Thai Unknown Words from the Web," In *Proc. of the COLING/ACL on Main Conference Poster Sessions*, pp. 345–352, 2006.

[3] C. Kruengkrai and H. Isahara, "A Conditional Random Field Framework for Thai Morphological Analysis," In *Proc. of the Fifth Int. Conf. on Language Resources and Evaluation (LREC-2006)*, 2006.

[4] T. Kudo, K. Yamamoto, and Y. Matsumoto, "Applying Conditional Random Fields to Japanese Morphological Analysis," In *Proc. of EMNLP*, pp. 230–237, 2004.

[5] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," In *Proc. of the Eighteenth Int. Conf. on Machine Learning (ICML)*, pp. 282–289, 2001.

[6] F. Peng, F. Feng, and A. McCallum, "Chinese Segmentation and New Word Detection Using Conditional Random Fields," In *Proc. of the 20th COLING*, 2004.

[7] V. Sornlertlamvanich, "Word Segmentation for Thai in Machine Translation System," *Machine Translation*, National Electronics and Computer Technology Center, Bangkok, 1993.

Measuring Search Engine Quality in Image Queries in 10 Non-English Languages: An Exploratory Study

Fotis Lazarinis
Technological Educational Institute of Mesolonghi
30200 Mesolonghi, Greece
lazarinf@teimes.gr

Efthimis N. Efthimiadis
The Information School, University of Washington
Seattle, WA, USA
efthimis@u.washington.edu

ABSTRACT
The main aim of this short paper is to measure the relevance of the retrieved images in a number of queries in different non-English languages. The reasoning behind retrieval failures is investigated and suggestions for improving these problems are discussed.

Categories and Subject Descriptors
H.3. [**Information Systems**]: Information Search and Retrieval
H.3.5 Online Information Services
H5.4: Hypertext/Hypermedia. – User issues.

General Terms
Measurement, Experimentation, Human Factors.

Keywords
Search engines, Image retrieval, Non-English Web retrieval.

1. INTRODUCTION
The number of images stored on the Web increases on an exponential rate and will continue to expand as the storage media become cheaper. The major search engines offer image searching facilities based on textual queries. Search engines utilize the text that surrounds the image or its file name to index and retrieve images [4]. Although the procedure of searching the Web for images is straightforward it has been reported that in non-English textual queries the accuracy of the produced results is reduced [8].

The main aim of this paper is to measure the accuracy of the retrieved images in a number of queries in different non-English languages. The reasoning behind retrieval failures is also investigated and suggestions for improving these problems are presented.

2. METHODOLOGY
To realize whether search engines produce quality results in non-English queries a number of queries were searched in ten (10) European languages. Table 1 presents the languages and their alphabets. Most of the Latin based languages of the sample contain additional characters to the ones used in English, most of which are special characters with diacritics, e.g., ä in German, ź in Polish, č in Croatian, å in Norwegian. This selection of languages is representative of the European languages because they use different scripts and are spoken by populations of varied sizes. For example, Greek and Norwegian are spoken by only a few million people while French and Spanish are spoken by several millions across the world. The queries were constructed in the following forms: (a) singular, (b) plural, (c) capitalized, and (d) without diacritics.

3. RESULTS and DISCUSSION
3.1 Queries in Singular Form
Table 2 presents the textual queries used in this evaluation. These queries were extracted from a search log that was obtained from four Greek academic institutions. The five most recent one-word image queries were used. The selection of one-word queries is based on the results of a study by Jansen and Spink which reported that European users have been searching the web using up to 35% of one-word queries [3] (p. 256).. The original queries were in Greek. Table 2 presents the queries used in lower case English and in the 10 non-English languages of our experiment. The translations from Greek to the other languages were completed with the aid of three web-based translation services and dictionaries[1] in order to cover all the languages of the sample and cross validate the accuracy of the translations.

Table 1. Languages used in the experiments

Language	Alphabet	Language	Alphabet
English	Latin	Russian	Cyrillic
French	Latin	Croatian	Latin
German	Latin	Spanish	Latin
Greek	Greek	Norwegian	Latin
Italian	Latin	Turkish	Latin
Polish	Latin		

These queries were run in lower case in both Google's and Yahoo's image search services; where possible the localized interfaces of these search engines were used. Google has localized searching interfaces for all the non-English languages presented in Table 1. For example, Google.gr and Google.pl are the Greek and Polish versions of Google respectively. Yahoo offers localized versions of the search interface only for Spanish, French, German, Italian, Russian and Norwegian. For the remaining four languages (Croatian, Greek, Polish, and Turkish) the http://images.yahoo.com version was used.

Table 3 reports the precision at the top 20 retrieved images for

[1] The three online translation services and dictionaries used were: http://babelfish.yahoo.com/, http://translate.google.com/, and www.tranexp.com:2000/Translate/result.shtml

each query and each search engine (P@20). It also reports the average precision over all queries per search engine, and the Mean Reciprocal Rank (MRR). MRR was calculated as the mean of the reciprocal of the rank at which the first relevant image retrieved across all queries. Zero was assigned if no correct image was retrieved in top 20 results. Relevance was assessed by visually inspecting the retrieved images.

Table 2. One word queries in lower case

Language	Q1	Q2	Q3	Q4	Q 5
English	dog	cat	computer	fish	flower
French	chien	chat	ordinateur	poissons	fleur
German	hund	katze	computer	fische	blume
Greek*	σκύλος	γάτα	υπολογιστής	ψάρι	λουλούδι
Italian	cane	gatto	computer	pesce	fiore
Polish	pies	kot	komputer	ryba	kwiat
Russian	собака	кот	компьютер	рыбы	цветок
Croatian	pas	mačka	računalo	riba	cvijet
Spanish	perro	gato	computador	pez	flor
Norwegian	hunden	katten	computer	fisken	blomst
Turkish	köpek	kedi	bilgisayar	balık	çiçek

* Diacritics are used in lower case Greek queries

Table 3. Relevance assessments of lower case image queries

Language	Search engine	Q1 dog	Q2 cat	Q3 computer	Q4 fish	Q5 flower	Avg precision	MRR
English	Google	100%	95%	60%	95%	95%	89%	0.90
	Yahoo	100%	100%	85%	85%	100%	94%	0.90
French	Google	95%	100%	95%	95%	100%	97%	0.90
	Yahoo	85%	85%	80%	95%	100%	89%	0.87
German	Google	100%	95%	70%	100%	100%	93%	1.00
	Yahoo	100%	95%	85%	85%	100%	93%	0.90
Greek	Google	100%	90%	85%	80%	100%	91%	1.00
	Yahoo	100%	95%	20%	55%	100%	74%	0.73
Italian	Google	95%	85%	80%	35%	90%	77%	0.90
	Yahoo	20%	100%	85%	25%	90%	64%	0.87
Polish	Google	90%	95%	85%	85%	100%	91%	1.00
	Yahoo	0%	5%	85%	15%	95%	40%	0.46
Russian	Google	90%	100%	90%	100%	100%	96%	1.00
	Yahoo	95%	90%	15%	50%	100%	70%	0.65
Croatian	Google	100%	100%	95%	90%	90%	95%	1.00
	Yahoo	5%	85%	35%	0%	95%	44%	0.50
Spanish	Google	100%	95%	100%	95%	95%	97%	1.00
	Yahoo	85%	95%	80%	10%	100%	74%	0.81
Norwegian	Google	60%	90%	80%	70%	95%	79%	0.87
	Yahoo	75%	20%	85%	25%	90%	59%	0.72
Turkish	Google	100%	100%	90%	95%	100%	97%	1.00
	Yahoo	100%	95%	40%	25%	95%	71%	0.66

As seen in Table 3, in several cases Yahoo failed to retrieve relevant images in the less spoken non-English languages. For example, Yahoo returned too few relevant results for Polish, Croatian, Norwegian and Turkish. By examining these failures it appears that Yahoo does not take into account the input language in searching. Therefore in cases where a non-English word, like "pies" (dog) in Polish, is used in searching, Yahoo does not retrieve images relevant to these languages. On the contrary Google, based on its localized interface, seems to the user to understand the input language and retrieves images that match the user's query. The reason that Google works better is because the localized version of the user interface restricts results to that domain, hence the increased accuracy in the results.

Another finding from this experiment is that it is important to use the localized interface of search engines in order to allow them to customize the results. For example, if we use the standard .com version of Google to retrieve images using the Croatian query "pas" (dog) then the results contain a lot of not relevant images while the Croatian interface retrieves 20 relevant images.

Based on the data in Table 3 it seems that the spoken languages which are based on the Latin alphabet are more error prone in image searching, that is the disambiguation of homonyms is a challenge. For example, the same word may have different meanings across Latin based languages and the search engines may retrieve images relevant to another meaning than the user's intentions. In Russian and Greek which are based on the Cyrillic and Greek scripts respectively, the alphabet itself seems to have a discriminatory power and therefore search engines seem to return better results.

Studying the filenames of the images retrieved in the Latin queries of our sample we found that most of the filenames contained the query text. For example, in the English query "dog" all the image filenames contained the text "dog". Also the URL of the image contained the word "dog". The same applies for the Italian query "cane" (dog). On the contrary, this technique is not applicable on the non-Latin queries. For example, none of the filenames contains the query text in the Greek query "σκύλος" (dog). Most of the filenames in this case contain the English term "dog" or the transliterated text "skylos". Therefore, image searching on non-Latin languages can only be based on the surrounding text and on the image's alternative text which should be thoroughly described to increase the chances of an image to be retrieved.

3.2 Queries in Capital Letters

The singular-form queries presented in Table 2 were transformed to upper case and were submitted again to Google and Yahoo. No significant differences were found in terms of accuracy in the top 20 ranked images in all queries but Greek. Diacritics are not used in upper case Greek queries. For exanmple, the query "σκύλος" (dog) is transformed to "ΣΚΥΛΟΣ" in upper case. These two queries retrieve different number of images in Google and although the accuracy is very high in both cases the overlap among the first 20 results is 75%. In Yahoo the lower case version produces 546 images and the upper case query retrieves 163 images. The correlation among the results in these two cases is very low. Further, the accuracy of the queries searched in upper case is quite low, i.e., only 8 out of the 20 top images present a dog in the upper case queries whereas all top 20 images of the query "σκύλος" are relevant to the query. However, this is mainly a problem related to diacritics and not to capital letters.

3.3 Queries in Plural Form

The next stage of this exploratory study was to run some image queries in plural form. The first two image queries of Table 2 were transformed into plural (see Table 4) and were run again in Google and Yahoo. Table 5 reports precision on the top 20 results (P@20), the average precision, and the Mean Reciprocal Rank (MRR). Yahoo's performance is not as good as Google's especially in Greek, Polish, Croatian, Russian and Turkish. These results are consistent with the results reported for lower case. For instance, queries like the Croatian "psi" (dogs) and the Polish "psy" (dogs) are either regular words in English or acronyms. Therefore, the search engines retrieve images related to the English search terms and not to the Polish or Croatian.

Table 4. One word queries in plural.

Language	Q1	Q2
English	dogs	cats
French	chiens	chats
German	hunde	katzen
Greek	σκύλοι	γάτες
Italian	cani	gatti
Polish	psy	koty
Russian	собаки	коты
Croatian	psi	mačke
Spanish	perros	gatos
Norwegian	hundene	kattene
Turkish	köpekler	kediler

Table 5. Relevance assessments of lower case image queries in plural form

Language	Search engine	Q1 dogs	Q2 cats	Avg precision	MRR
English	Google	100%	95%	98%	1
	Yahoo	95%	95%	95%	1
French	Google	95%	100%	98%	1
	Yahoo	100%	100%	100%	1
German	Google	95%	95%	95%	1
	Yahoo	100%	85%	93%	1
Greek	Google	85%	95%	90%	1
	Yahoo	65%	100%	83%	0.75
Italian	Google	95%	100%	98%	1
	Yahoo	95%	100%	98%	1
Polish	Google	95%	100%	98%	0.75
	Yahoo	0%	40%	20%	0.525
Russian	Google	100%	95%	98%	1
	Yahoo	40%	95%	68%	0.75
Croatian	Google	70%	100%	85%	1
	Yahoo	0%	45%	23%	0.525
Spanish	Google	100%	100%	100%	1
	Yahoo	100%	85%	93%	1
Norwegian	Google	100%	75%	88%	1
	Yahoo	90%	95%	93%	1
Turkish	Google	100%	100%	100%	1
	Yahoo	60%	80%	70%	0.75

When comparing the results in Tables 3 and 5 it can be observed that in some cases the plural form of the words produced more relevant queries than the singular form. This is the case, for example, in the Norwegian searches for both Google and Yahoo. In a few other cases, like in the Turkish queries the plural form of the search terms, produce less relevant results than the singular form queries. Several of the non-English languages are highly inflectional due to the different inclinations of nouns, adjectives and verbs. Based on this and on the observation about different levels of accuracy between the singular and plural query forms, search engines should become aware that the same query might be expressed in different forms. This suggests that stemming could solve such problems.

In non-Latin languages, like Greek, transliteration of text into Latin is a common practice in computer mediated communications [5]. This also applies to filenames used for images. For example, Greek users store images about "γάτες" (cats) using the transliterated text "gates". When users search for "γάτες" (cats) using Google.gr, results containing images with the word "gates" in their filenames are returned. As pointed in [6] this technique produced more results but the relevance is not increased. Google seems to combine regular queries with the transliterated text in an attempt to retrieve better results. However, this may lead to erroneous instances and therefore should be used only in combination with other information. For instance, an image of Bill Gates is also retrieved in the query "γάτες" (cats) which is transliterated to "gates". The filename of this image of Bill Gates is "bill-gates-off-facebook" and its part "gates" is emboldened in the results.

3.4 Queries with Diacritics

All the non-English languages used in the study contain some letters with diacritics in their alphabets. For instance, the French and Spanish alphabets include the "è" and "á" letters in their alphabets. In Polish and Croatian even consonants, e.g. "ź" and "č", may have diacritics. Further, all Greek vowels may take an accent mark. Table 6 presents some one-word queries with diacritics and their English equivalents which were randomly chosen from respective dictionaries.

Table 6. One word queries with diacritics

Language	Word with diacritics	Word in English
French	crème	cream
German	känguru	kangaroo
Greek	σκύλος	dog
Italian	università	university
Polish	łódź	a Polish city
Russian	ёжики	hedgehog
Croatian	mačka	cat
Spanish	tulipán	tulip
Norwegian	gård	farm
Turkish	köpek	dog

The purpose of this part of the experiment was to understand whether search engines handle efficiently the absence of diacritics from textual queries. The motivation behind this part of the experiment originated from previous studies which researched the presence or absence of diacritics from non-English Web queries and the fact that more often than not users type their Web queries without diacritics [1][2][7].

Before running the experiment, the queries of Table 6 were submitted to Google Web Search limiting the results to the respective language. The queries were submitted without diacritics and the purpose was to see whether these words appear in Web pages without diacritics. All the queries returned a respectable number of Web pages proving that in several occasions, words are typed without diacritics in Web documents.

Next, the queries were searched with and without diacritics in both Google and Yahoo. Table 7 presents the relevance assessments in both cases. As seen, Google has comparable performance in both cases. Yahoo, on the other hand fails to retrieve relevant images in most queries where diacritics are omitted. In Greek and Russian it even fails to retrieve a number of images. One might argue that handling the versions with and without diacritics equally might not be appropriate. However, as previous studies showed users neglect to include diacritics in their queries [8]. Further, in several occasions diacritics are not used in filenames or in capital letters. Therefore search engines should be aware of these problems and handle queries without diacritics more effectively.

Table 7. Relevance of queries with and without diacritics

Language	Search engine	Query with diacritics	Query without diacritics	Avg precision	MRR
English	Google	100%	95%	98%	1
	Yahoo	90%	90%	90%	1
French	Google	95%	95%	95%	1
	Yahoo	90%	30%	60%	1
German	Google	100%	100%	100%	1
	Yahoo	100%	25%*	63%	1
Greek	Google	80%	75%	78%	1
	Yahoo	80%	80%	80%	0.75
Italian	Google	100%	90%	95%	1
	Yahoo	95%	75%	85%	1
Polish	Google	100%	100%	100%	0.75
	Yahoo	85%	0%**	43%	0.525
Russian	Google	100%	100%	100%	1
	Yahoo	85%	5%	45%	0.75
Croatian	Google	100%	95%	98%	1
	Yahoo	100%	100%	100%	0.525
Spanish	Google	90%	20%	55%	1
	Yahoo	100%	0%	50%	1
Norwegian	Google	100%	95%	98%	1
	Yahoo	100%	15%	58%	1
Turkish	Google	100%	95%	98%	1
	Yahoo	90%	90%	90%	0.75

* 6 images were retrieved
** 0 images were retrieved

4. CONCLUDING REMARKS

Image searching is a daily activity of millions of Web users. The major search engines offer image searching facilities based on textual queries. Motivated from previous studies which discuss the problems in non-English web queries, this research studied a number of one-word image queries in 10 non-English languages to discover potential problems.

One of the main findings of this paper is that the input language is important as it helps the search engine to disambiguate the query. Image filenames are encoded in Latin scripts even in non-Latin languages. Therefore, search engines cannot match the search terms with the filenames in non-Latin languages. In this manner, users of non-Latin languages have fewer chances to retrieve relevant images than Latin encoded queries. On the other hand, Latin queries in less popular languages, like in Croatian, have a higher chance of retrieving worse results because query terms may represent a different concept in a major Latin based language, as demonstrated with the example of "pies" (dogs).

Efficient handling of semantically similar but morphologically different queries is important since several non-English languages are highly inflectional. Users may submit their queries in plural or in singular forms or in variable declination cases (e.g., nominative, accusative, etc.). Search engines should be aware of the grammatical idiosyncrasies of complex non-English languages to produce more quality results. The same applies for search terms with diacritics. Since users may omit them from queries or from filenames search engines should be able to efficiently handle them.

The main conclusion of this pilot study is that image searching in non-English and non-Latin languages is less successful than in English queries. Search engines should become more aware of the linguistic problems of non-English languages and of how users type their queries in order to retrieve more relevant images among the top ranked queries.

To validate these results and further investigate the shortcomings of non-English web search for images more testing is needed with a larger number of queries as well as with queries consisting of multiple words. Search engines should provide alternative queries and translations to English to help users retrieve relevant images in case their original queries fail. The ability to narrow the results to the national domains may improve the response of search engines, like Yahoo, and help users satisfy their information needs.

5. REFERENCES

[1] Choroś, K. 2005. Testing the Effectiveness of Retrieval to Queries Using Polish Words with Diacritics. Advances in Web Intelligence Conference, LNCS 3528, DOI= 10.1007/b137066

[2] Efthimiadis, E. N., et al. 2008. An Evaluation of How Search Engines Respond to Greek Language Queries. HICSS-41: IEEE Hawaii International Conference on System Sciences, Waikoloa, Big Island, Hawaii.

[3] Jansen, B.J. and Spink, A. 2006. How are we searching the World Wide Web? A comparison of nine search engine transaction logs. Information Processing and Management, 42, 248–263

[4] Kherfi, M. L., Ziou, D., Bernardi, A. 2004. Image Retrieval from the World Wide Web: Issues, Techniques, and Systems. ACM Computing Surveys (CSUR) archive, 36(1), 35 – 67

[5] Koutsogiannis, D. and Mitsikopoulou B. 2003. Greeklish and Greekness: Trends and Discourses of "Glocalness". *Computer-Mediated Communication*, 9(1).

[6] Lazarinis, F. 2007. An initial exploration of the factors influencing retrieval of Web images in Greek queries. EATIS07 International Conference, DOI=http://doi.acm.org/10.1145/1352694.1352765

[7] Lazarinis, F. 2007. How do Greek searchers form their Web queries? 3rd International Conference on Web Information Systems and Technologies, (WEBIST), Spain, 404-407.

[8] Lazarinis, F. 2008. Improving concept-based web image retrieval by mixing semantically similar Greek queries. Program: electronic library and information systems, 42(1), pp. 56-67, DOI=10.1108/00330330810851

Issues in Searching for Indian Language Web Content

Dipasree Pal, Prasenjit Majumder, Mandar Mitra, Sukanya Mitra, and Aparajita Sen
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute, Kolkata
{dipasree_t,prasenjit_t,mandar,sukanya_t,aparajita_t}@isical.ac.in

ABSTRACT

This paper looks at the problem of searching for Indian language (IL) content on the Web. Even though the amount of IL content that is available on the Web is growing rapidly, searching through this content using the most popular web-search engines poses certain problems. Since the popular search engines do not use any stemming / orthographic normalization for Indian languages, recall levels for IL searches can be low. We provide some examples to indicate the extent of this problem, and suggest a simple and efficient solution to the problem.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Experimentation

Keywords

Indian Language

1. INTRODUCTION

The last decade has witnessed the increasing proliferation of the Internet in India. With more and more people accessing the World Wide Web in the country, the amount of Indian language content available in electronic form has reached critical mass. A large number of newspapers have online editions. There are also several magazines and information portals serving content in various Indian languages. Table 1 shows the number of news related sites listed in the Google Directory (directory.google.com) for three major ILs. The Indian parliament publishes proceedings of its sessions[1]. The Government is also encouraging the introduction of local languages in official activities. With all this information available online, there is a growing need for techniques for effectively managing the available information.

[1]http://www.parliamentofindia.nic.in/

Language	# sites
Bengali	29
Hindi	54
Tamil	65

Table 1: News-related sites listed in Google Directory

Unfortunately, in many cases, directly using the existing search systems on IL text collections does not yield very good results. In this paper, we highlight two specific drawbacks that we have identified in popular search engines. We provide some anecdotal evidence to indicate the extent of the problem, and suggest a simple approach to tackle this problem.

2. INDIAN LANGUAGE CONTENT ON THE WEB

One of the most important issues that continue to come in the way of rendering IL Web content searchable is the use of proprietary encodings [1,4]. Even though the use of Unicode is becoming more widespread, IL content is still frequently represented in terms of font codes rather than character codes. In other words, a character is represented by the numerical code(s) for the glyph(s) that make(s) up the visual representation of that character. The set of glyphs and their numerical codes could be different for different fonts for the same language. Even worse, some online publications provide articles in the form of document images, and other multimedia formats (such as Macromedia Flash).

Another issue that arises when considering the problem of IL search is the lack of a single orthographic standard. Different newspapers have their own conventions with regard to spelling. This is specially true for proper nouns which are named entities, and which are likely to occur often as search terms.

This problem is compounded by the fact that a substantial part of the content is created in informal settings, such as blogs and newsgroups. Such content is often multi-lingual, and written using the Roman alphabet, and since there is no single, universally accepted transliteration scheme, the same words may appear in different transliterated forms in different sources.

3. EXPERIMENTS

To get a better understanding of the extent of the problems outlined above, we conducted an empirical study. First, we assess the distribution of different encoding schemes used to display IL texts. A list of online daily publications in Hindi and Bengali was collected from the Google directory. For each site, we examined the content encoding scheme used. Table 2 shows that although most Hindi online resources use UTF-8, the use of proprietary, non-standard encodings continues to remain a problem for Bengali, the second most widely spoken IL. Incidentally, we observe that most Bengali blogs are written either in UTF-8 or using the Roman script.

Language	# sites	UTF	Prop. fonts	Images, flash, etc.
Bengali	29	9	14	6
Hindi	54	33	9	12

Table 2: Content encoding in Bengali and Hindi

Next, we collected the most commonly used search terms[2] from Google's India site (http://google.co.in). These terms were translated into Bengali and Hindi and submitted to four search engines, namely Google, Yahoo, Altavista and Guruji, a search engine developed with the explicit goal of indexing IL texts. Table 3 shows the number of hits for the various search terms across various search engines. This table confirms our hypothesis that the amount of information available online in some of the major ILs has crossed critical mass.

Search terms	Google	Yahoo	AltaVista	Guruji
Wikipedia				
(Bengali)	153,000	638,000	639,000	90,958
(Hindi)	256,000	2,710,000	2,720,000	31,556
Mobile				
(Bengali)	87,500	91,400	91100	5,612
(Hindi)	53,700	119,000	115,000	3,004
Hanuman				
(Bengali)	6,270	1,260	1260	434
(Hindi)	160,000	122,000	121,000	5,658
Harry Potter				
(Bengali)	766	236	249	159
(Hindi)	10,900	5,420	5,390	1,182
BSNL				
(Bengali)	134	77	81	25
(Hindi)	31,100	35,800	36,100	3,967
Statue of Liberty				
(Bengali)	17	5	6	3
(Hindi)	36	5	6	4

Table 3: Hits generated by popular search terms across search engines

Next, we study the impact of morphological variations. A set of Bengali terms, along with morphological variations, was submitted to the search engines. It was observed that a substantial number of the relevant documents is missed by the engine when only the root word is given. Table 4 lists the number of pages returned for a word and its variations. This clearly indicates the importance of a stemmer for a morphologically rich language like Bengali. This problem is much less serious for Hindi, which is less inflectional in nature.

Finally, we studied the extent of spelling variations in Bengali and Hindi. Due to the existence of several dialects and inclusion of loan words form European and Middle Eastern languages, some words (especially names) have several valid spellings. Tables 5 and 6 list some variants for commonly occurring names that are likely to be used as search terms.

[2]http://www.google.com/press/zeitgeist.html

4. CONCUSIONS

The preceding section clearly shows that stemming / orthographic normalization is necessary in order to increase the effectiveness of IL searches. On the other hand, given the volume of documents and queries that search engines have to handle, any normalization scheme has to be simple and efficient in order to be useful in practice. A simple stemmer could be constructed by creating a list of the most commonly used suffixes, and deleting matching suffixes from the ends of words (provided the remaining portion is longer than a certain minimum length). This approach was used successfully by Oard et al. [3] at CLEF 2000. An overview of other statistical stemming approaches can be found in [2].

Spelling variations in Indian languages arise primarily due to the existence of (i) long and short forms of certain vowels, (ii) three sibilants, and (iii) two forms of the nasal "n". Normalization may be achieved by mapping all variants in each category to a single, canonical form, i.e. long vowels may be mapped to the corresponding short forms, all sibilants may be mapped to a single character, etc. Such rules have been successfully used in the past [4].

In future work, we intend to quantitatively evaluate the impact of these normalization techniques on search effectiveness.

5. REFERENCES

[1] Prasenjit Majumder, Mandar Mitra, and Bidyut B. Chaudhuri. An OCR-based architecture for indexing Indian language web documents. In *Proceedings 2nd Symposium on Indian Morphology, Phonology and Language Engineering (SIMPLE 05)*, 2005.

[2] P. Majumder, M. Mitra, S.K. Parui, G. Kole, P. Mitra, and K. Datta. YASS: Yet Another Suffix Stripper. *ACM Transactions on Information Systems*, 25(4), October 2007.

[3] D.W. Oard, G.-A. Levow, and C.I. Cabezas. CLEF Experiments at Maryland: Statistical Stemming and Backoff Translation. In *Proc. CLEF Workshop on Cross-Language Information Retrieval and Evaluation*, pages 176–187, Springer-Verlag, 2001.

[4] Prasad Pingali, Jagadeesh Jagarlamudi, and Vasudeva Varma. Webkhoj: Indian language IR from multiple character encodings. In *Proceedings of WWW2006 Workshop*, May 2006.

Word	Inflected forms			
village	গ্রাম 52900	গ্রামের 14000	গ্রামে 12600	Others 572
Europe	ইউরোপ 72700	ইউরোপের 11200	ইউরোপীয় 4720	Others 6882
America	আমেরিকা 55500	আমেরিকায় 2710	আমেরিকার 7850	Others 1415
democracy	গণতন্ত্র 6490	গণতান্ত্রিক 5490	-	Others 288
deep	গভীর 32800	গভীরতা 1270	গভীরতর 648	Others 361
Olympic	অলিম্পিক 17300	অলিম্পিকে 612	-	Others 837
NASA	নাসা 637	নাসার 481	নাসায় 451	Others 6
NATO	ন্যাটো 930	ন্যাটো-র 570	-	Others 22
Shoaib	শোয়েব 1520	শোয়েবের 108	শোয়েবকে 377	Others 5
IPL	আইপিএল 261	আইপিএল-এ 91	আইপিএল-এর 64	Others 17

Table 4: Page hits for a word and its morphological variations (Bengali)

Word	Variations					Total
English	ইংলিশ 12,500	ইংরাজি 31,200	ইংরেজি 98,200	ইংরিজি 261	-	1,42,161
Asia	এশিয়া 73,300	এসিয়া 22,100	-	-	-	95,400
Hindi	হিন্দি 16,700	হিন্দী 61,400	-	-	-	78,100
Sachin	শচীন 31,700	সচিন 790	শচিন 619	-	-	33,109
Orissa	ওড়িশা 866	উড়িষ্যা 24,200	ওড়িষ্যা 3,940	-	-	29,006
arrest	গ্রেফতার 8,350	গ্রেপ্তার 5,690	-	-	-	14,040
community	কম্যুনিটি 6,910	কমিউনিটি 6,030	-	-	-	12,940
Israel	ইজরায়েল 2,080	ইসরাইল 3,830	ইজরাইল 192	ইসরেল 75	ইসরাঈল 1,410	7,587
database	ডেটাবেজ 1,370	ডাটাবেজ 1,400	ডাটাবেস 1,830	ডেটাবেস 1,770	-	6,370
Tendulkar	টেন্ডুলকার 359	তেন্ডুলকার 73	তেন্ডুলকর 134	-	-	566

Table 5: Page hits for variant spellings (Bengali)

Word	Variations			Total
petrol	पैट्रोल 5,840	पेट्रोल 2,67,000	-	2,72,840
New Delhi	नई दिल्ली 3,23,000	नयी दिल्ली 48,800	-	3,71,800
rupees	रुपए 40,700	रुपये 3,20,000	-	3,60,700
companies	कंपनियां 84,800	कम्पनियां 9,610	-	94,410
Bahubali	बाहुबली 7,980	बाहुबलि 93	-	8,073
couple	दम्पति 10,000	दंपत्ति 6,560	-	16,560
agency	एजेंसी 367,000	एजेन्सी 16,500		3,83,500
diesel	डीजेल 1,82,000	डीज़ेल 23,300	-	2,05,300
Valentine	बैलेंटाइन 6,330	बेलेंटाइन 9,480	बेलेन्टाइन 486	16,296

Table 6: Page hits for variant spellings (Hindi)

Baseline for Urdu IR Evaluation

Kashif Riaz
University of Minnesota
4-192 EE/CS Building
200 Union Street SE
Minneapolis, MN 55455
riaz@cs.umn.edu

ABSTRACT
Goal of conferences like TREC, TIPSTER, NTCIR, CLEF is to judge the performance of different algorithms. Most of these conferences have tracks that deal with new and innovative information retrieval problems, but none has tackled to work with Urdu data, primarily because of the lack of resources. In this paper we present a baseline for Urdu IR evaluation along with resources necessary to do the task. The goal of this paper is to explore the strategy for creation of the test reference collection for Urdu Information Retrieval. A small test reference collection is presented composed of news articles from the Web along with some experiments to show its effectiveness.

Categories and Subject Descriptors
H.3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness).

General Terms
Experimentation, Languages, Measurement

Keywords
Urdu, Evaluation, Baseline, Relevance Judgments, Test Reference Collection

1. INTRODUCTION
There have been tremendous advances in the Information Retrieval (IR) community since the field emerged about forty years ago [1]. But unfortunately, until the last decade almost all research was done in English. Since then the community has made some advances in other European and Asian languages, but still there has been little effort done for languages that are written from right to left with the exception of Arabic.

There is a need to study and build Information Retrieval technology in languages that although spoken by a large number of world's population have not gotten attention in the IR community primarily because of the lack of language resources like corpora and expertise; right to left languages is an example of such languages. Most of these languages with a few exceptions like Hebrew, were and still are transliterated on the Web or represented as images because of the lack of Unicode support of the Operating Systems, or the in the presentation software. An Urdu word can be transliterated into many ways. For example, world's most common first name Mohammad, can be transliterated as at least a dozen different ways, Muhammad, Mohamed, and Mohd to list a few. While trying to search for a non-English name like Mohammad, one has to query a search engine multiple times to get all the relevant materials, but it can only be written one way in Urdu, Farsi and Arabic. A brief explanation of some characteristics of Urdu is provided in Section 2.

Our larger goal is to build an Urdu Concept Search engine. In order to accomplish this we need a robust evaluation mechanism. Retrieval Performance [2] evaluation is based on a reference test collection and on an evaluation measure. The reference collection consists of a test corpus, a set of queries, and set of relevant document for each query. Queries are constructed by the domain experts and the relevant documents are selected by experts. Query generation experts and relevant document selectors need not be the same. We provide a reference collection and query relevance judgments for 200 Urdu documents from the Becker-Riaz Urdu corpus [3]. This is quite a challenging task for Urdu because of the dearth of specialists who understand the technicalities of the task at hand, which is creating a test reference collection and its usefulness in Information Retrieval discipline. Creation of the Urdu baseline is explained in section 4. We follow the TREC methodology in order to create our baseline. This approach has been consistent across most new initiatives for example INEX 2007 Book Search task [5]. We briefly describe TREC evaluation in section 3.

There are many measures proposed in the IR literature to judge the performance of an IR system. For some measures, ranking of the documents is baked in, while others do not use ranking information to judge the performance. In our experiments we use both types of measures to baseline the Urdu IR system. Some of the widely used measures are: Recall-Precision, F-measure, Mean Average Precision, and R-Precision [2]. Although, not the focus of this paper, we check operability of our reference collection with Boolean searching and Ranked Retrieval (Vector Space Model) techniques to measure the Recall and Precision of the documents. The test results are shown in section 5.

2. URDU
We briefly introduce some right to left languages and a few characteristics of Urdu. Recently, there has been quite a bit of interest in right to left language processing in the IR community, specifically in the intelligence community and other organizations working for the government in the United States. Most of the interest has been focused toward Arabic (a right to left language).

There are other right to left language like Urdu, Persian (Farsi), Dari, Punjabi, and Pashto that are mostly spoken in South Asia. Arabic is a Semitic language and the other languages belong to the Proto Indo Iranian branch of languages. Arabic and these other languages only share script and some vocabulary. Therefore, the language specific task done for Arabic is not applicable to these languages.

Urdu is the national language of Pakistan, and one of the major languages of India. It is estimated that there are about 300 million speakers of Urdu. Most of the Urdu speakers live in Pakistan, India, UAE, U.K and USA. Urdu is considered the *lingua franca* of business in Pakistan, and the South Asian community in the U.K [4]. Urdu among all above languages mentioned has unique case that it shares its grammar with Hindi; the difference is some vocabulary, and writing style. Hindi is written in Devanagari script. Therefore, Hindi and Urdu are considered one language for linguistic purposes. Urdu is quite complex language because its morphology is a combination of many languages: Sanskrit, Arabic, Farsi, English and Turkish to name a few. This aspect of Urdu becomes quite a challenge while doing morphological analysis to build a stemmer. Urdu's descriptive power is quite high. This means that there could be many different ways a concept can be mentioned in Urdu and in many different forms. For example, the words *Pachem* and *Maghreb* both are used for the direction *West.* In the previous example *Pachem* has its ancestry in Sanskrit and *Maghreb* has its roots in Arabic.

Urdu has a property of accepting lexical features and vocabulary from other languages, most notably English. This is called code-switching in linguistics e.g. it is not uncommon to see a right to left flow interrupted by a word written in English (left to right) and then continuation of the flow right to left. For example,وہ میرا laptop ہے [That is my laptop]. In the above example, Microsoft Word did not support English embedding within the Urdu sentence and displayed it improperly. But while electronically processing the tokenization will be done correctly [3].

3. TREC TEST REFERENCE COLLECTION

The TREC collection, also called the TIPSTER collection, was initiated under the leadership of Donna Harman [13] at National Institute of Standards and Technology (NIST) Conference. The goal was to have the collection that consisted of millions of documents, provide a uniform scoring procedures and forum for organizations who are interested in comparing their results [2]. TREC has test reference collections for many languages but not for Urdu.

The search results of the participant systems are run through the same evaluation system so consistent evaluation results are seen and the participants can compare and contrast results with each other. The TREC conference has many different tracks, like Ad hoc, novelty, routing etc. Like test collections, the TREC test bed consists of three essential parts: the documents in the test collection (corpus), the example queries, these are called *topics* in the TREC domain, and a set of relevant documents for each example query. TREC document collection (corpus) is composed of various genres like news articles, computing reviews and legal documents to mention a few. All documents in the collection are tagged with SGML like tags. There is no stemming done on the words and stop words have not been removed. Some statistics are provided for each genre, an example statistics for a genre is given in Table 1.

Table 1

Genre	Size in MB	#Docs	Median Words/Doc	Mean Word/Doc
Financial Times 1991-1994	564	210,158	316	412.7

A skeleton of TREC document is given below:

```
<DOC>
<DOCNO>document_number</DOCNO>
<TEXT>
document_text
</TEXT>
</DOC>
```

There could be other tags present in other sub-collections or the type of the data e.g. new articles have the tag of <h1> to indicate the title, <author>, and <dateline>. The details for each format can be obtained from TREC's Web site [8]

The example queries or topics in TREC nomenclature are created by experts who have long been in the information seeking business. The query (topic) is represented in an SGML-like snippet. An example topic from TREC 2006 Ad hoc track given below

```
<top>
<num> Number: 301
<title> International Organized Crime
<desc> Description:
 Identify organizations that participate in international criminal
activity, the activity, and, if possible, collaborating organizations
and the countries involved.
<narr> Narrative:
  A relevant document must as a minimum identify the organization and
the type of illegal activity (e.g., Columbian cartel exporting cocaine).
Vague references to international drug trade without identification of the
organization(s) involved would not be relevant.
</top>
```

The format and the comprehensiveness of the topics have varied over various TREC conferences. Since TREC corpus contains millions of documents it is unrealistic to provide relevance judgments for each query, therefore, a pooling method is used to judge the relevant documents. Pooling method combines the top *N* results for each participant system, an intersection of these results are shown to judges for evaluation. Pooling method has to be quite effective and empirically correct.

4. URDU TEST REFERENCE COLLECTION

One of the major hurdles for Urdu processing is the lack of baseline evaluation mechanism for results. There has been some work done on other Indian language evaluation through Forum for Information Retrieval Evaluation (FIRE) [9]. In this section we address creating a gold standard (baseline) for Urdu that is typically available for other languages. We followed the Ad hoc track in order to build our Urdu baseline.

Information retrieval is fundamentally a user driven task where the search engine is trying to satisfy an information need of a user. This is evident from users reissuing queries after unsatisfactory results. In the absence of a test collection one can always do user studies for evaluation. But this is an expensive proposition in the absence of funding. So we decided to use the TREC methodology in order to create our baseline. We soon realized the in the absence of any funding, and most importantly the lack of human resources to create the queries and relevance judgments we cannot replicate the work done by TREC on the same scale. In our case we have two Urdu corpora available, the EMILLE corpus (300 documents) [6] and the Becker-Riaz corpus (7000 documents) [3]. EMILLE is not suited for creating relevance judgments and queries, because it contains documents that are quite large in size, discuss quite disjointed topics and are quite unlike Web documents. Becker-Riaz corpus consists of Web news stories, and therefore could be used for generating relevance judgments and queries. Unfortunately, there are few researchers in the IR community who know Urdu and IR concepts and most importantly could volunteer to create topics and relevance judgments. We chose 200 documents for the test corpus because it is quite manageable to create relevance judgments and create topics and supervise the work produced by topic and relevance judgment creators. We were able to get two volunteers from Pakistan and one from the United States to generate queries and relevance judgments. The resources from Pakistan were university students and the resource from the United States was a lifelong newspaper reader who had a good command of the politics and news in South Asia. The resource in the United States was quite naive about the use of computers and the field of Information Retrieval, so it required quite a bit of coaching before giving the task, and scrutinizing of work after the deliverable.

The documents in our test reference collections were kept in Corpus Encoding Standard (CES) [7], a departure from TREC format. We chose this format because Becker-Riaz corpus is metadata rich and we plan on using this metadata for our algorithm development for the larger goal for concept search.

The topics were constructed in the same fashion as TREC format, but the snippet is well-formed XML instead of SGML-like snippet in TREC which is not well-formed XML. The title information of the topic is in Urdu, but other metadata like description of the topic are in English. An example of topic is given below:

```
<topic>
<number>1</number>
<urdu.title> کرکٹ کپ </urdu.title>
<english.title>Cricket Cup</english.title>
<description> Identify articles that discuss the news stories about different
cricket tournaments, opinions about the matches and results and
performance of batsmen and bowlers. The documents could be about the
World Cup or other cricket tournaments where a number of cricket
playing countries participate for a prize </description>
<narration> A relevant document must as a minimum identify the
document that relate to cricket tournaments where a major prize is
awarded </narration>
</topic>
```

The relevance judgments are provided in a table where the set of documents that match the topic are listed. Some statistics of the 200 Urdu documents are given below.

Table 2

Genre	Size MB	#Docs	Median Words/Doc	Mean Word/ Doc
Becker-Riaz	1	200	119	183

5. RESULTS

In order to check the operability of the test reference collection, we evaluated a home-grown Boolean retrieval engine, Apache Lucene [10], Lemur [11], and Terrier [12] search engines. Lemur (a C++ based engine) had encoding and XML processing issues. When document was transformed into TREC format to circumvent its XML processing issues, Lemur indexed Urdu fine, but retrieval did not recognize the query. The transformation of XML documents into TREC like format is not desirable because a lot of useful metadata information is lost that way from the Becker-Riaz corpus e.g. keywords, author, date, word count, and type of document (news story, column etc.). We found Lucene to be the most Unicode, XML and metadata aware search engine. We have not yet completed our evaluation with Terrier because of the lack of documentation and examples in an organized manner.

The words in the corpus were not stemmed and stop words were not removed from the documents. The purpose of keeping the raw corpus was to establish a baseline and then use stemming and stop word removal techniques to measure the Recall and Precision of the same queries. Our results showed that the Boolean engine works accurately with logical OR and logical AND queries in isolation, but was not working well with the combination of AND-OR queries. Lucene is a combination of Boolean and Vector Space Model and has fielded search capability. We did our experimentation with out-of-the box Lucene engine without any weights on terms. We used three types of queries or topics to establish our baseline: simple queries, concept queries, and phrase queries. Simple queries are one word queries, concept queries can be multiword and are designed to retrieve documents that may or may not contain the keyword, and phrase queries contain Urdu phrases. Some results of each query type are given below.

We used the simple query پرل *(Pearl)* to search for news articles about the journalist Daniel Pearl. Initial testing showed that the fielded search on *Title* retrieved about 6 of 11 relevant documents, but all the retrieved documents were relevant. The *Para* fielded search resulted in all the 11 documents. When we used the query اسرائیل *(Israel)* 6 out of 25 relevant documents were retrieved when we used *Title* fielded search. The results missed a number of documents because the query was not اسرائیلی *(Israeli)*. These two terms will conflate to one after stemming and provide a much better recall. The results show that our relevance judgments can be used to identify the reasons for low recall as shown in the query *Israel.* We used a concept query پاکستانی عدلیہ (*Pakistan's Judicial System)* to retrieve news stories about situation of Pakistan's court system. There were no relevant documents retrieved when we used *Para* fielded search. The *Title* fielded search retrieved 1 of the 4 relevant documents. This is mostly because Lucene is a keyword-based engine. For phrase queries we used the phrase نیپال میں ماؤنواز باغیوں کی سرگرمیاں *(Activities of Maoist rebels in Nepal)*. This query had 3 relevant documents but the *Title* fielded search retrieved 104 documents and the *Para* fielded search retrieved 109 documents. This is because Lucene did a logical-OR of all the query terms including the two stop words in the query. When the stop words were removed, the *Title* fielded search retrieved 4 documents 3 relevant and 1 non relevant about rebels in Fiji. The *Para* fielded search retrieved 6 documents of which 3 were relevant. The results from the phrase query suggest that the relevance judgments can be used to identify the importance of stop word removal to improve precision and recall of the IR system.

Given the results, we think that our baseline provides good support of Urdu IR evaluation. It is important to note that we do not show the results of extensive experiments or create recall-precision curves because the goal of this paper is to present the test reference collection not the results of the experiments using the baseline.

6. CONCLUSION & FUTURE WORK

We have shown that in the absence of a robust test reference collection for Urdu, a small test reference collection can be used in order to establish a baseline for Urdu processing algorithms. Becker-Riaz corpus is best suited for the Web retrieval algorithms because it is comprehensively composed of news articles from the Web. Besides measures like recall and precision, we intend to use F-measure, novelty-ratio and coverage ratio [2]. We want to use Kappa statistics across different users to determine the variability of judgments across experts. Eventually, a TREC like *pooling* will be used to find the common relevant documents across Lucene, Terrier and the home grown engine. Also, we will like to increase our test reference collection to at least a 1000 documents.

7. REFERENCES

[1] Singal; “Modern Information Retrieval” IEEE Data Engineering, 2001

[2] R. Bates-Yates, B. Ribeiro-Neto, “Modern Information Retrieval”, Addison Wesley, 1999

[3] D. Becker, K. Riaz. “A Study in Urdu Corpus Construction.” Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization at the 19th International Conference on Computational Linguistics. August 2002.

[4] P. Baker, A. Hardie, T. McEnery, and B.D. Jayaram. “Corpus Data for South Asian Language Processing”. Procee-dings of the 10th Annual Workshop for South Asian Language Processing, EACL 2003.

[5] K. Gabriella, A. Doucet, "Overview of the INEX 2007 Book Search Track (BookSearch '07)", SIGIR Forum June 2008, vol. 42, no. 1 pp 2-15.

[6] W0038: The EMILLE Lancaster Corpus. http://www.elda.org/catalogue/en/text/W0038.html, (July 2008)

[7] N. Ide, C. Brew. “Requirements, Tools, and Architectures for Annotated Corpora”. Proceedings of Data Architectures and Software Support for Large Corpora. European Language Resources Association, Paris, 2000.

[8] http://trec.nist.gov/ (July 2008)

[9] P. Majumder, M. Mitra, S. K. Parui, P. Bhattacharyya. The First International Workshop on Evaluating Information Access (EVIA 2007) Tokyo, Japan, May 15, 2007.

[10] Lucene. http://lucene.apache.org/ (July 2008)

[11] The Lemur Toolkit for Language Modeling and Information Retrieval. http://www.lemurproject.org/ (July 2008)

[12] Terrier http://ir.dcs.gla.ac.uk/terrier/ (July 2008)

[13] E. Voorhees, D. Harman: TREC Experiment and Evaluation in Information Retrieval MIT Press, Cambridge, 2005

Cross-Lingual Query Classification: a Preliminary Study

Xuerui Wang*
Univ. of Massachusetts
140 Governors Drive
Amherst, MA 01003, USA
xuerui@cs.umass.edu

Andrei Broder Evgeniy Gabrilovich Vanja Josifovski Bo Pang
Yahoo! Research
2821 Mission College Blvd.
Santa Clara, CA 95054, USA
{broder, gabr, vanjaj, bopang}@yahoo-inc.com

ABSTRACT

The non-English Web is growing at breakneck speed, but available language processing tools are mostly English based. Taxonomies are a case in point: while there are plenty of commercial and non-commercial taxonomies for the English Web, taxonomies for other languages are either not available or of very limited quality. Given that building taxonomies in all non-English languages is prohibitively expensive, it is natural to ask whether existing English taxonomies can be leveraged, possibly via machine translation, to enable information processing tasks in other languages. Preliminary results presented in this paper indicate that the answer is affirmative with respect to query classification, a task which is essential both for understanding the user intent and thus providing better search results, and for better targeting of search-based advertising, the economic underpinning of commercial Web search engines. We propose a robust method for classifying non-English queries against an English taxonomy using widely available, off-the-shelf machine translation systems. In particular, we show that by viewing the search results in the query's original language as independent sources of information, we can alleviate the impact of poor quality or erroneous machine translations. Empirical results for Chinese queries show that we achieve remarkably encouraging results.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Machine translation*

General Terms

Algorithms, Measurement, Experimentation

1. INTRODUCTION

The Web has grown rapidly since its inception in 1992 at an approximately exponential growth rate. Initially, most of the Web content was in English; however, as more users go online worldwide, the importance of the non-English part of the Web increases steadily. While English still dominates the Web, in 2002 as many as 43.6% of the Web content was in languages other than English, and the percentage of non-English queries submitted to Google was reported to have increased from 36% to 43% over a 6-month period.[1] Furthermore, Web usage in non-English-speaking countries has exploded in the last decade. For example, according to *Internet World Stats*[2], as of March 2008, China had about 210 million Internet users, second only to the United States that had 218 million.

Despite the increasing importance of the non-English Web, significantly fewer technical resources are available in those languages. Developing such technical resources for each language of interest to us can be an extremely labor-intensive and expertise-intensive task. It is therefore of great interest to apply resources already available in English to processing other languages instead of developing such resources anew for each language.

One natural direction to achieve this aim is to use automatic machine translation systems. While the field of machine translation (MT) has advanced significantly over the recent years, it is still not feasible to depend on MT systems to reliably translate training examples (let alone develop entire taxonomies) into the target language, owing to the less-than-perfect quality of MT output. Instead, we use MT systems to provide an admittedly *imperfect* mapping between English and non-English languages, and use MT output as an intermediate step that undergoes further processing. It is this indirect use of machine translation that allows our system to tolerate translation errors.

In this paper, we focus on query classification, where most of the previous work was conducted for the English Web. Query classification has proven to be effective for better understanding query intent and improving user experience, as well as for boosting the relevance of online advertising [2, 3]. For instance, knowing that the query "TI-83" is about graphical calculators while "E248WFP" is about LCD monitors can obviously lead to more focused advertisements even though no advertiser has specifically bid on these particular queries. A commercial English taxonomy of Web queries with approximately 6000 nodes where each node was populated with example queries has been developed in previous work [3]. Translating this taxonomy into each non-English language of interest to us and re-populating the translated

*This work was performed during the author's summer internship at Yahoo! Research in 2008.

iNEWS'08, October 30, 2008, Napa Valley, California, USA.
Copyright 2008 ACM 978-1-60558-253-5/08/10 ...$5.00.

[1]http://www.netz-tipp.de/languages.html
[2]http://www.internetworldstats.com/top20.htm

taxonomy with example queries in that non-English language can be very labor-intensive. Instead, we classify non-English queries with respect to the original English taxonomy by utilizing classifiers built for English text directly. The labels can be used to improve the algorithmic results for non-English Web search as well as the quality of advertisement placement in non-English languages.

A straight-forward way to classify a non-English query is to directly translate the query into English, and use existing techniques for English query classification. However, while machine translation tools work reasonably well on longer text fragments, they can be quite inaccurate on very short text such as typical Web queries. Consequently, inaccurate translation can cause the subsequent classification to go completely astray, which can no longer be corrected even with additional resource on the English side.

In this paper we propose a more robust method for classifying non-English queries. Instead of directly translating a query into English, we submit the query to a search engine, machine translate and classify the resulting pages, and then infer the query class from the page classes. We present preliminary experimental results on queries sampled from a Chinese query log. We show that significantly better classification accuracy can be obtained via our approach compared to directly translating queries.

2. RELATED WORK

Recently, there has been a surge of interest in cross-language text classification. Classification results over various language pairs have been reported, including, but not limited to, English-Italian [9], English-Czech [8], English-Spanish [6], English-Japanese [4], and English-Chinese [7]. Bel et. al [1] discuss two main approaches to cross-language text classification: *poly-lingual training*, where a classifier is trained on labeled training documents in multiple languages, and *cross-lingual training*, where a classifier is trained in one source language, and documents in other languages are completely or selectively translated into the source language for classification. Our method bares more resemblance to the second approach.

Query classification can be considered as a special case of text classification in general, but it is in a sense much more difficult due to the brevity of queries. Observe, however, that in many cases a human looking at a search query and the search results does remarkably well in making sense of it. Unfortunately, the sheer volume of search queries does not lend itself to human supervision, and alternative sources of knowledge about the world are needed. The state-of-the-art method [3] uses a blind relevance feedback technique: given a query, the class label is determined by classifying the Web search results retrieved for the query. Empirical evaluation confirms that this procedure yields a considerably higher classification accuracy than previous methods.

In this paper, we approach the task of non-English query classification by taking advantage of advances in both cross-language classification and query classification. To the best of our knowledge, none of previously published work has addressed this important, and extremely difficult problem.

3. METHOD

We present a method for classifying non-English queries with respect to an English taxonomy with the help of external knowledge in the query's native language. Given a query, we first submit it to a Web search engine and retain a few top-scoring search results; we then translate these search results into English using a machine translation system. The translated results are subsequently classified using an existing classifier trained on English data. Finally, we perform voting among the predicted classes of individual search results to determine the class(es) for the original query. The overall procedure of our method is shown in Figure 1:

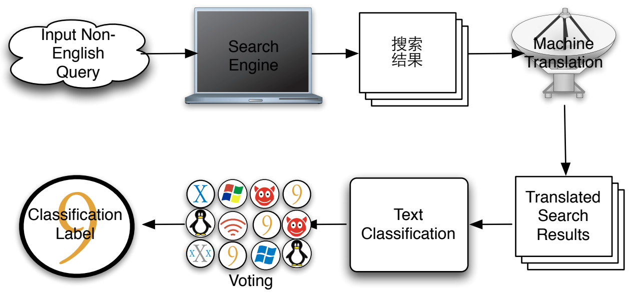


Figure 1: Robust classification of non-English queries.

Web Search.

First, we dispatch a given non-English query to one or more major search engines to retrieve top N search results in the query's native language.

In this study, queries are dispatched to Google to retrieve up to 32 search results (due to the limit imposed by Google AJAX Search API). The top search results are crawled from the Web using the returned URLs. When a fresh copy is not available, Google's cached page is retrieved with Google's cache header removed to ensure that these pages are comparable to the original pages.

All crawled Web pages are processed to remove all the tags, java scripts, and other non-content information. If the returned results are not HTML files (e.g., PDF files, MS Word documents, etc.), they are simply removed from consideration. The resulting non-English textual content is re-encoded into UTF-8 regardless of the original encoding.

Machine Translation.

The crawled Web pages are translated into English via an off-the-shelf machine translation system. To study the impact of using different MT systems, we experiment with two different systems that are easily accessible over the Web:

- *Babel Fish*[3] is powered by the technology of SYSTRAN, one of the oldest machine translation companies, that at least at one point was known to be a rule-based MT system.
- *Google Translate*[4] is said to be based on statistical machine translation techniques with which the system is trained on large-scale parallel corpora.

The public interfaces of both systems have limits on query length. We break long texts into parts, translate them separately, and merge the translations afterwards.

[3]http://babelfish.yahoo.com
[4]http://translate.google.com

Text Classification.

The translated pages are classified into an English taxonomy by a centroid-based classifier [5] trained on English data. This classifier has been shown to be efficient and effective for large-scale experiments. Up to 5 ranked labels are returned for each page.

Label Voting.

Finally, we infer the query class from the page classes. More specifically, we take the top 5 classes with the most votes from the page class predictions as the most likely class labels of the original query, with each translated page contributing up to 5 votes with equal weights.

Compared to the baseline approach of direct query translation, our method has three advantages. First, by dispatching the original query to a search engine, we expand the query with exogenous knowledge that would not be available otherwise. In particular, while the query itself might be difficult to translate (e.g., the name of a popular Chinese TV series), the search results will likely contain additional pertinent keywords indicative of the correct class label that are easier to translate. Second, state-of-the-art machine translation systems are much better at translating long Web pages than short queries, thus considerably reducing the amount of erroneous translations introduced by the MT system. Even though the translated Web pages might not be easily readable by human readers, a machine-learned classifier can still reliably classify MT output [7]. Finally, the voting mechanism further increases the robustness of our method as it alleviates the impacts of irrelevant search results or partially incorrect translations. The ranking of search results also gives us the flexibility to experiment with weighted voting procedures.

4. EXPERIMENTAL RESULTS

In this section, we first describe our data set; we then evaluate our approach using two different machine translation systems and compare the results to the baseline approach that directly translates queries.

4.1 Data Set

We apply our method to 200 queries sampled from a large-scale Chinese query log. It is well known that the volume of queries in today's search engines roughly follows a power law. To make the 200 samples representative of the overall traffic, we divide the query log into ten deciles with respect to the logarithm of query frequency, and sample 20 queries uniformly from each decile. This way, we ensure both popular and rare queries are represented in our sample.

4.2 Evaluation Mechanism

Preliminary pilot studies show that directly classifying machine translated queries yields extremely poor results due to the poor quality of machine translation of short queries. To further strengthen the baseline, we employ the blind relevance feedback procedure by expanding the translated English query with search results in the English Web, similar to what was done in state-of-the-art English query classification systems [3]. Note that this enhanced baseline approach is quite powerful in itself with the help of external evidence from the Web. Thus, both the proposed approach and the baseline system take advantage of machine translation as well as the blind relevance feedback techniques. However, by doing Web search first in the original language, we significantly increase the robustness of our approach since machine translation errors are reduced or partially compensated for as more relevant content is available in the native language.

For each system under comparison, we take up to 5 predicted labels for each query. Since there is no existing ground truth of Chinese query labels, two native Chinese speakers were asked to make editorial judgments over each predicted label into *correct* (1) or *incorrect* (0). To remove possible bias towards any particular approach, all predicted labels from different systems are mixed and presented to the human editors in random order.

As different editors can have different interpretations of the original query intent, their judgments can slightly differ from time to time. We define the correctness of a prediction in two ways: the logical AND (both judges consider the label as correct) or the logical OR (one of the judges considers the label as correct) of the two judgments. For each query, the performance of a particular method is measured by the percentage of correct predictions among the top 5 predicted labels. Note that although we refer to this performance measure as accuracy in the text to follow, it is not accuracy per se: a query might have only two correct labels, in which case even a perfect classifier is bounded by 40% accuracy with this measure. Still, this measure demonstrates the relative effectiveness of different approaches under consideration, as more comprehensive comparisons using different metrics are not included due to space limit.

4.3 Results

We report performance of four different systems: the proposed method and the enhanced baseline method paired up with Google Translate or Babel Fish, denoted as *Method+Google*, *Method+Babelfish*, *Baseline+Google*, and *Baseline+Babelfish*, respectively.

The average accuracies over queries sampled from different deciles are shown in Table 1, where Decile 1 corresponds to the most frequent queries, and Decile 10 corresponds to the least frequent queries. Performances measured by using logical AND (OR) to combine editorial judgments are presented in the top (bottom) part of the table.

We first compare the performance of the proposed method against that of the corresponding baseline system using the same MT system, and mark the corresponding number with a superscript when our method significantly outperforms the baseline under one-tail paired t-test with p-value< 0.05: "+" for *Method+Google* vs. *Baseline+Google*; and "$\diamond$" for *Method+Babelfish* vs. *Baseline+Babelfish.* As shown in Table 1, regardless of which translation system is used, the proposed method outperforms the baseline approach most of the time. We conjecture that, with more queries, the performance difference will be much more significant. Note that for less frequent queries, the performance gap between our method and the baseline method becomes larger, which probably reflects the difficulty of translating rare queries. Given the queries are sampled from different volume deciles and is therefore somewhat representative for the overall traffic, we conjecture that this improvement will reasonably carry over to larger sets of sample queries. In the future, a more comprehensive larger-scale experiment will enable us to draw stronger conclusions.

Table 1: The average accuracy over queries, with the top part corresponding to logical AND of editorial judgments, and the bottom part corresponding to logical OR of editorial judgments. The superscripts denote significance under one-tail paired t-test with p-value< 0.05.

Decile	**Method+ Google**	**Method+ Babelfish**	**Baseline+ Google**	**Baseline+ Babelfish**
1	0.470	$0.480^{\diamond}$	0.440	0.370
2	0.470^{+}	$0.440^{\diamond}$	0.290	0.190
3	0.350^{+}	$0.340^{\diamond}$	0.180	0.180
4	0.320	0.290	0.270	0.300
5	0.380^{+}	$0.390^{\diamond}$	0.170	0.160
6	$0.410^{\star}$	0.340	0.310	0.250
7	0.410^{+}	$0.350^{\diamond}$	0.080	0.100
8	0.320^{+}	$0.290^{\diamond}$	0.220	0.190
9	0.420^{+}	$0.360^{\diamond}$	0.210	0.180
10	0.270	0.250	0.240	0.220
Overall	$0.382^{\star+}$	$0.353^{\diamond}$	0.241	0.214
1	0.620	0.610	0.620	0.560
2	0.620^{+}	$0.610^{\diamond}$	0.440	0.260
3	0.520^{+}	$0.480^{\diamond}$	0.310	0.250
4	0.550^{+}	0.510	0.400	0.380
5	0.570^{+}	$0.530^{\diamond}$	0.310	0.250
6	0.610	$0.530^{\diamond}$	0.480	0.380
7	$0.550^{\star+}$	$0.440^{\diamond}$	0.170	0.130
8	$0.440^{\star}$	0.370	0.350	0.290
9	0.610^{+}	$0.560^{\diamond}$	0.340	0.300
10	0.470^{+}	0.430	0.380	0.350
Overall	$0.556^{\star+}$	$0.507^{\diamond}$	0.380	0.315

We also examine the effect of using different MT systems. "$\star$" denotes significant difference between *Method+Google* and *Method+Babelfish*. We observe that, overall, *Method+Google* significantly outperforms *Method+Babelfish* on the 200 queries. In the future, we plan to apply our approach with a simple bilingual-dictionary-based tranlsation module to further investigate how the quality of machine translation affects the performance of our system.

5. CONCLUSIONS AND DISCUSSIONS

In this paper, we presented a robust method to classify non-English queries against an English taxonomy. We dispatch a non-English query to a general purpose search engine, and retrieve top search results in the query's native language. These non-English pages are then translated into English via publicly available MT systems. The translated pages are classified using a classifier trained on English data, and the label of the given query is inferred from the classes of the translated pages by a voting mechanism.

Preliminary experiments with queries sampled from a Chinese query log show that our method almost always significantly outperforms a strong baseline method, and the conclusion holds consistently for two different machine translation systems. By employing the blind relevance feedback techniques in the query's native language, rather than in the English Web with the translated query, the impact of erroneous translation is significantly reduced.

It is also important to note that the performance of our method, as we expect, seems to be less sensitive to the volume of the query, that is, less query dependent, with an overall smaller variance in average accuracy over different deciles, and relatively better performance on rare queries compared to the baseline approach. The rare queries, in aggregation accounting for a considerable mass of the search engine traffic, simply do not have enough occurrences to allow statistical learning on a per-query basis. The superior performance of our method on rare queries provides a substantial opportunity for down-stream applications such as online advertising.

While a relatively small query set is used in our preliminary study, the results are quite promising. In the future, we plan to further investigate the robustness of our method with larger data sets in multiple non-English languages.

6. REFERENCES

[1] N. Bel, C. H. A. Koster, and M. Villegas. Cross-lingual text categorization. In *Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries*, pages 126–139, 2003.

[2] A. Broder, P. Ciccolo, M. Fontoura, E. Gabrilovich, V. Josifovski, and L. Riedel. Search advertising using Web relevance feedback. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, 2008.

[3] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 231–238, 2007.

[4] A. Gliozzo and C. Strapparava. Exploiting comparable corpora and bilingual dictionaries for cross-language text categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 553–560, 2006.

[5] E.-H. Han and G. Karypis. Centroid-based document classification: Analysis and experimental results. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 424–431, 2000.

[6] Y. Li and J. Shawe-Taylor. Advanced learning algorithms for cross-language patent retrieval and classification. *Information Processing and Management*, 43(5):1183–1199, 2007.

[7] X. Ling, G.-R. Xue, W. Dai, Y. Jiang, Q. Yang, and Y. Yu. Can chinese web pages be classified with english data source? In *Proceeding of the 17th international conference on World Wide Web*, pages 969–978, 2008.

[8] J. S. Olsson, D. W. Oard, and J. Hajič. Cross-language text classification. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 645–646, 2005.

[9] L. Rigutini, M. Maggini, and B. Liu. An EM based training algorithm for cross-language text categorization. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 529–535, 2005.