Practical NLP-Based Text Indexing

J. Vilares, F. M. Barcala, M. A. Alonso, J. Graña, and M. Vilares

Departamento de Computación, Universidade da Coruña
Campus de Elviña s/n, 15071 La Coruña, Spain
{jvilares,barcala}@mail2.udc.es {alonso,grana,vilares}@udc.es
http://coleweb.dc.fi.udc.es/

Abstract. We consider a set of natural language processing techniques based on finite-state technology that can be used to analyze huge amounts of texts. These techniques include an advanced tokenizer, a part-of-speech tagger that can manage ambiguous streams of words, a system for conflating words by means of derivational mechanisms, and a shallow parser to extract syntactic-dependency pairs. We propose to use these techniques in order to improve the performance of standard indexing engines.

1 Introduction

In recent years, there has been a considerable amount of interest in using Natural Language Processing (NLP) in Information Retrieval (IR) research, with specific implementations varying from the word-level morphological analysis to syntactic parsing to conceptual-level semantic analysis. In this paper we consider the employment of a set of practical NLP techniques built on finite-state technology that make them adequate for dealing with large amounts of texts. Finite-state technology is sometimes characterized as ad-hoc. However, we propose a sequence of finite-state based processes, where each stage corresponds to intuitive linguistic elements, reflecting important universals about language:

- The existence of individual words and idioms forming each sentence.
- The existence of different categories of word carrying the semantics of the sentence: nouns, adjectives and verbs.
- The existence of semantic relations between words belonging to different categories (e.g. the noun corresponding to the action of a verb).
- The existence of basic syntactic structures relating words within a sentence, such as the noun-modifier, subject-verb or verb-object relations.

The scheme of the paper follows the processing stages shown in Fig. 1. Firstly, in Sect. 2, we describe the preprocessor, an advanced tokenizer which accounts for a number of complex linguistic phenomena, as well as for pre-tagging tasks. Section 3 shows the tagger, which is based on Hidden Markov Models with disambiguation and lemmatization capabilities. Next, in Sect. 4, we describe the

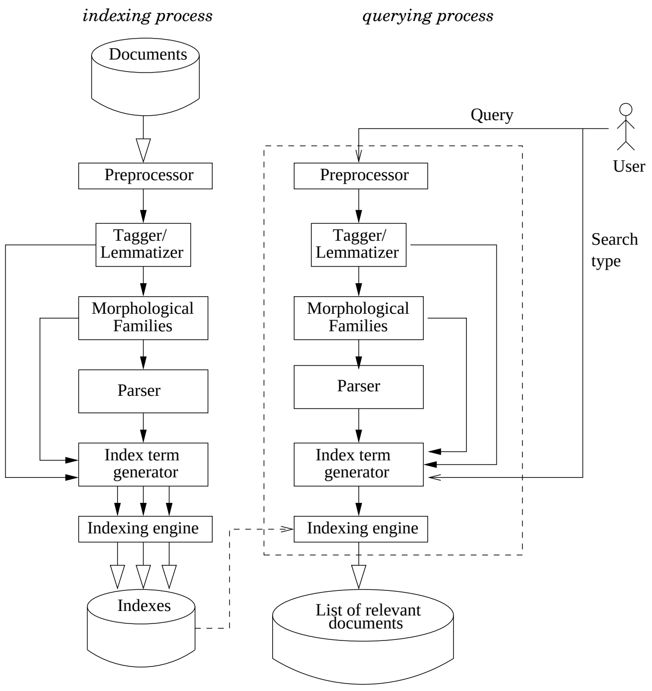

Fig. 1. General architecture of the system

main morphological mechanisms of word formation, and their application to the automatic generation of morphological families. Section 5 describes a shallow parser working on syntactic and morpho-syntactic variants of noun phrases for the extraction of syntactic dependency pairs. The evaluation of the proposed techniques is performed in Sect. 6. Section 7 presents final conclusions.

2 The Preprocessor

Current taggers assume that input texts are correctly segmented in *tokens* or high level information units that identify every individual component of the texts. This working hypothesis is not realistic due to the heterogeneous nature of the application texts and their sources. For this reason, we have developed a preprocessor module [4, 1], an advanced tokenizer which performs the following tasks:

Filtering. Texts are converted from source format (e.g. HTML or XML) to plain text, and delimiters are compacted (e.g. it removes multiple blanks or blanks at beginning of sentences).

Tokenization. Every individual word as well as every punctuation mark will be a different token, taking into account abbreviations, acronyms, numbers with decimals and dates in numerical format. For this purpose, we use two dictionaries, one of abbreviations and another one of acronyms, and a small set of rules to detect numbers and dates.

Sentence Segmentation. The general rule consists of separating a sentence when there is a dot followed by a capital letter. However, it must be taken into account certain abbreviations to avoid marking the end of a sentence at their dots.

Morphological Pretagging. The preprocessor tags elements whose tag can be deduced from the morphology of the word, and there is no more reliable way to do it. In this step, for instance, numbers and dates are identified.

Contraction Splitting. Contractions are split into their different tokens, assigning a tag to every one of them, by using external information on how contractions are decomposed. For instance, the Spanish contraction `del` (*of the*) is decomposed into the preposition `de` (*of*) and the article `el` (*the*).

Enclitic Pronouns. Verb stems are separated from their enclitic pronouns, tagging every one of them correctly. To perform this function, we need to consult a dictionary with as many verbal forms as possible, a dictionary containing the greatest possible number of verbal stems capable of presenting enclitic pronouns, a list with all the valid combinations of enclitic pronouns, and a list with the whole set of enclitic pronouns, together with their tags and lemmas. As an example, the Spanish word `comerlo` (*to eat it*) is decomposed in `comer` (which is the infinitive *to eat*) and `lo` (which is the pronoun *it*).

Expression Identification. The different tokens that make up an expression are joined together [2], using a dictionary with the expressions that are uniquely expressions, e.g. `a pesar de` (*in spite of*), and a dictionary of phrases that may be expressions or not, e.g. `sin embargo` (*however* or *without seizure*). The preprocessor simply generates the possible segmentations, and then the tagger selects one of those alternatives later.

Numeral Identification. Consecutive numerals are joined together in order to build a compound numeral and so obtain only one token. For instance, every component of `mil ciento veinticinco` (*one thousand one hundred and twenty-five*) is joined with the rest in the same way as the components of an expression. Unlike the case of expressions, the tag assigned by the preprocessor here is definitive.

Proper Noun Training. Given a sample of the texts that are going to be indexed, the preprocessor identifies the words that begin with a capital letter and appear in non-ambiguous positions, i.e. in positions where if a word begins with a capital letter then it is a proper noun. For instance, words appearing after a dot are not considered, and words in the middle of the text are considered. It also identifies sequences of capitalized words connected by some valid connectives like the preposition *of* and definite articles. The proper nouns detected are added to a trained dictionary.

Proper Noun Identification. Using a specific dictionary of proper nouns and the trained dictionary, we are able to detect proper nouns whether simple or compound, and either appearing in positions ambiguous or not. This task is explained in detail in [1].

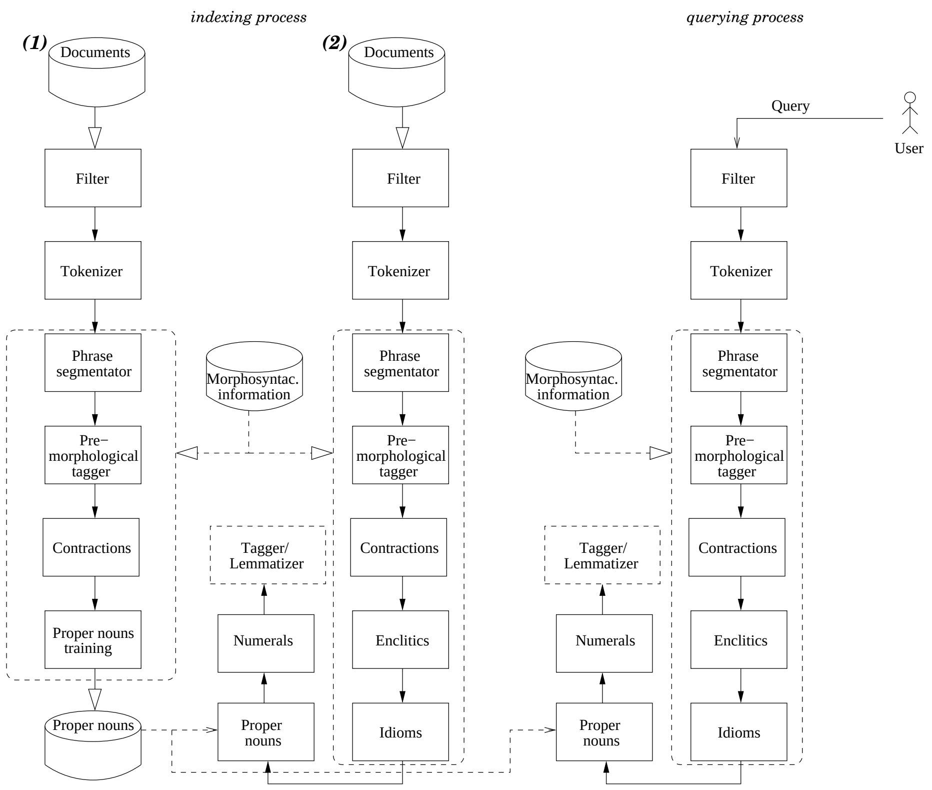

Fig. 2. General architecture of the preprocessor

The general structure of this first module is shown in Fig. 2. As we can see, there are two processing modes. Firstly, there is an off-line process, during indexing time, where the documents to be indexed are tagged. This off-line process consists of two steps. In the first step, a subset of the documents is used for proper noun training, and in a second step, the data obtained are employed to tag the entire document database. The other main processing mode is an on-line process during querying time where the query is tagged. The data obtained in the proper noun training phase during indexing process is also employed here for tagging the query.

3 The Tagger

A second order Hidden Markov Model (HMM) is used to perform part-of-speech tagging. The states of the model represent pairs of tags, and outputs represent the words. Transition probabilities depend on the states, thus pairs of tags. Output probabilities only depend on the most recent category. To be explicit, we use the Viterbi algorithm to calculate:

$$\arg\max_{t_1 \ldots t_n} \prod_{i=1}^{n} [P(w_i|t_i) \times P(t_i|t_{i-2}, t_{i-1})]$$

for a given sentence of words $w_1 \ldots w_n$ of length n, where $t_1 \ldots t_n$ are elements of the tagset. Transition and output probabilities are estimated from a tagged corpus. As a first step, we use the maximum likelihood probabilities derived from relative frequencies. As a second step, contextual frequencies are smoothed and lexical frequencies are completed by handling words that do not appear in the training corpus but are present in external dictionaries.

Trigram probabilities generated from a corpus cannot be used directly because of the sparse-data problem, which means that there are insufficient instances for each trigram to reliably estimate the probability. The smoothing paradigm that delivers the best results is linear interpolation of unigrams, bigrams and trigrams. Therefore, we estimate a trigram probability as follows:

$$P(t_3|t_1, t_2) = \lambda_3 \hat{P}(t_3|t_1, t_2) + \lambda_2 \hat{P}(t_3|t_2) + \lambda_1 \hat{P}(t_3)$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$, so P again represents probability distributions. The values of λ_1, λ_2 and λ_3 are estimated by deleted interpolation.

Given an unknown word, its candidate tags and their probabilities are set according to the ending of the word in question. The probability distribution of a particular suffix is generated from all words in the training set that share the same suffix of some predefined maximum length. Probabilities are then smoothed by successive abstraction [9]. This method obtains a proper probability distribution for each tag and for each suffix length, as needed by the HMM.

Sometimes we have to deal with languages with very few available linguistic resources. Currently, the typical situation in Spanish processing is very short training texts, but very large dictionaries, since the morphology of the language is well-known and the effort made to formalize it has been much greater. The most intuitive way to integrate a dictionary is the Adding One method, which consists of using the dictionary as an additional tagged corpus where a frequency of 1 is assigned to each word-tag pair. However, this integration does not produce a coherent representation of the model we are estimating, and it can produce important alterations in the working parameters. This leads us to consider another method based on the Good-Turing formulas [7]. Every word-tag pair present only in the external dictionary can be seen as an event with null frequency in the training corpus, and the Good-Turing formulas are themselves a method able to assign probabilities greater than 0 to these rare but existing events. In addition, this technique produces less distortion of the model and increases the performance of the tagging process when the training corpus is small [5].

Due to the ambiguous segmentations obtained during preprocessing, as it was described in Sect. 2, this tagger must be able to deal with streams of tokens of different lengths: it not only has to decide the tag to be assigned to every token, but also to decide whether some of them form or not the same term, and assign the appropriate number of tags on the basis of the alternatives provided by the preprocessor. To perform this process, we consider the evaluation of every stream of tokens and their subsequent comparison, in order to select the most probable one, as indicated in [3]. It is also necessary to define some objective criterion for that comparison. When the tagging paradigm used is the framework of the hidden Markov models, as is our case, that criterion is the comparison of the normalization of the cumulative probabilities. One reason to support the use of hidden Markov models is that, in other tagging paradigms, the criteria for comparison may not be so easy to identify.

Once a text has been tagged, content words (nouns, verbs, adjectives) are extracted to be indexed. In this way we solve the problems derived from inflection in Spanish. Therefore, recall is remarkably increased. With regard to computational cost, the running cost of a lemmatizer-disambiguator is linear in relation to the length of the word and cubic in relation to the size of the tagset, which is a constant. As we only need to know the grammatical category of the word, the tagset is small and therefore the increase in cost with respect to classical approaches (stemmers) becomes negligible.

4 Morphological Families

Once inflectional variation has been solved, the next logical step is to solve the problems derived from derivational morphology. Spanish has a great productivity and flexibility in its word formation mechanisms by using a rich and complex productive morphology, preferring derivation to other mechanisms of word formation. We define a *morphological family* as the set of words obtained from the same morphological root through derivation mechanisms. It is expected that a basic semantic relationship will remain between the words of a given family, relations of the type process-result, e.g. *producción* (production) / *producto* (product), process-agent, e.g. *manipulación* (manipulation) / *manipulador* (manipulator), etc. In Spanish, the basic derivational mechanisms are: *prefixation*, preposing morphemes to the base; *emotive suffixation*, postposing morphemes that alter the base in some sort of subjective emotional way; *non-emotive suffixation*, postposing morphemes that change the meaning of the base fundamentally rather than marginally, often effecting a change of syntactic category; *back formation*, a morphological procedure to derive nouns from verbs by truncation; and *parasynthesis*, the simultaneous prefixation and suffixation of the base lexeme.

Many derivational morphemes have variable forms (*allomorphs*), sometimes phonologically determined, and others lexically imposed by convention or etymology. It must also be taken into account that morphological operations can also involve phonological alterations of the base [8].

Regular word formation patterns in Spanish can be obtained through the 'rules of word formation' [8] defined by generative phonology and transformational-generative grammars. Though this paradigm is not complete, it has been used to implement an automatic system for generation of morphological families with an acceptable degree of completeness and correction [11].

Given two words w and w' in the lexicon, we denote by $w \triangleright w'$ the fact that w' is obtained from w by means of some of the derivational mechanisms shown above. Given this, we compute the morphological family of w as its reflexive and transitive closure through derivation, denoted closure(w) and defined recursively as:

- $w \in \text{closure}(w)$.
- If $w \triangleright w'$ then $w' \in \text{closure}(w)$.
- If $w' \triangleright w$ then $w' \in \text{closure}(w)$.

The set of morphological families associated with a given lexicon is obtained by means of applying closure(w) to each word w in the lexicon.

In order to use morphological families for document conflation, once we have obtained the part of speech and the lemmas of the text to be indexed, we replace each of the lemmas obtained by a fixed representative of its morphological family, which is indexed [11]. In this way we are using the same index term to represent all words belonging to the same morphological family; therefore, semantic relations that exist between these words remain in the index because related terms are conflated to the same index term. With regard to computational cost, morphological families and their representatives are computed *a priori*, so they do not affect the final indexing and querying cost.

5 The Shallow Parser

Given a stream of tagged words, the parser module tries to obtain the *head-modifier* pairs corresponding to the most relevant syntactic dependencies:*noun-modifier*, relating the head of a noun phrase with the head of a modifier; *subject-verb*, relating the head of the subject with the main verb of the clause; and *verb-complement*, relating the main verb of the clause with the head of a complement.

It has to be noted that while the head-modifier relation may suggest semantic dependence, what we obtain here is strictly syntactic, even though the semantic relation is what we are really after.

The kernel of the grammar used by the parser is inferred from the basic trees corresponding to noun phrases and their syntactic and morpho-syntactic variants [6, 10]:

Syntactic variants result from the inflection of individual words and from modifying the syntactic structure of the original noun phrase. Given a noun phrase, their syntactic variants are obtained by means of:

- *synapsy*, changing a preposition or adding or removing a determiner;
- *substitution*, employing modifiers to make a term more specific;
- *permutation* of words around a pivot element;
- employing *coordinating constructions* (copulative or disjunctive) with the modifier or with the modified term.

	stm	*lem*	*fam*	*sdp*
Average precision	0.2179	0.2214	0.2268	0.2931
Average recall	0.6335	0.6211	0.6397	0.5179

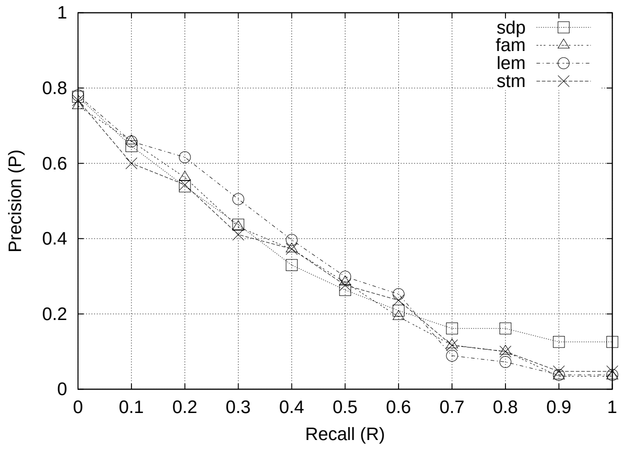

Fig. 3. Experimental results on a corpus of newspaper articles

Morpho-syntactic variants differ from syntactic variants in that at least one of the content words of the original noun phrase is transformed into another word derived from the same morphological stem. They can be classified according to the nature of the morphological transformations applied to their words:

– *Iso-categorial:* morphological derivation process does not change the category of words, but only transforms one noun syntagma into another. There are two possibilities: noun-to-noun and adjective-to-adjective.
– *Hetero-categorial:* morphological derivation does result in a change of the category of a word. There are also two possibilities: noun-to-verb and noun-to-adjective.

We must remark that syntactic variants involve inflectional morphology but not derivational morphology, whereas morpho-syntactic variants involve both inflectional and derivational morphology. In addition, syntactic variants have a very restricted scope (the noun phrase) whereas morpho-syntactic variants can span a whole sentence, including a verb and its complements.

Once the basic trees of noun phrases and their variants have been established, they are compiled into a set of regular expressions, which are matched against the tagged texts in order to extract the dependency pairs, which are used as index terms, as is described in [10]. In this way, we can identify dependency pairs through simple pattern matching over the output of the tagger/lemmatizer, dealing with the problem by means of finite-state techniques, leading to a considerable reduction of the running cost.

6 Evaluation

The lack of a standard evaluation corpus has been a great handicap for the development of IR research in Spanish.[1] This situation is changing due to the incorporation in CLEF-2001[2] of a Spanish corpus (composed of news provided by a Spanish news agency) which is expected to become a standard. The techniques proposed in this paper have been integrated recently, therefore, we could not participate in CLEF-2001 edition, but we are prepared to join competition in 2002. Due to the unavailability of the CLEF corpus, we have chosen to test our techniques over the corpus used in [12], formed by 21,899 newspaper articles (national, international, economy, culture,. . .) with an average length of 447 words. We have considered a set of 14 natural language queries with an average length of 7.85 words per query, 4.36 of which were content words.

The techniques proposed in this article are independent of the indexing engine we choose to use. This is because we first conflate the document to obtain its index terms; then, the engine receives the conflated version of the document as input. So, any standard text indexing engine may be employed, which is a great advantage. Nevertheless, each engine will behave according to its own characteristics (indexing model, ranking algorithm, etc.). We have compared the results obtained, using SMART with the `ltc-lnc` weighting scheme as indexing engine, by four different indexing methods: stemmed text after eliminating stopwords (*stm*), lemmatized text (*lem*), text conflated by means of morphological families (*fam*) and syntactic dependency pairs (*sdp*). Results are shown in Fig. 3. We can observe that *lem* and *fam* slightly improve the precision of *stm*, with *fam* also improving recall. With respect to *sdp*, we must remark it shows an improvement in precision of 34.5% with respect to *stm*.

7 Conclusion

In this article we have proposed a set of practical natural language techniques to improve the performance of text indexing when applied to Spanish texts. These techniques include an advanced tokenizer for the right segmentation of texts which accounts for a number of complex linguistic phenomena, a part-of-speech tagger based on a stochastic model that can manage ambiguous streams of words and integrate external dictionaries, a system for identifying words related by derivational morphology, and a parser to extract head-modifier pairs. All of them are built on finite-state technology, so they are very efficient and can be applied to tasks in which huge amounts of text need to be analyzed, as is the case of information retrieval. Albeit our scheme is oriented towards the indexing of Spanish texts, it is also a proposal of a general architecture that can be applied to other languages with very slight modifications.

[1] The test collection used in the Spanish track of TREC-4 (1995) and TREC-5 (1996), formed by news articles written in Mexican-Spanish, is no longer freely available.

[2] `http://www.clef-campaign.org`

Acknowledgements

Supported in part by Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica (TIC2000-0370-C02-01), Ministerio de Ciencia y Tecnología (HP2001-0044) and Xunta de Galicia (PGIDT01PXI10506PN).

References

1. Fco. Mario Barcala, Jesús Vilares, Miguel A. Alonso, Jorge Graña, and Manuel Vilares. Tokenization and proper noun recognition for information retrieval. In *3rd International Workshop on Natural Language and Information Systems (NLIS 2002), September 2-3, 2002. Aix-en-Provence, France*, Los Alamitos, California, USA, 2002. IEEE Computer Society Press.
2. Jean-Pierre Chanod and Pasi Tapanainen. A non-deterministic tokeniser for finite-state parsing. In *Proceedings of the Workshop on Extended finite state models of language (ECAI'96)*, Budapest, Hungary, 1996.
3. Jorge Graña, Miguel A. Alonso, and Manuel Vilares. A common solution for tokenization and part-of-speech tagging: One-pass Viterbi algorithm vs. iterative approaches. In *Text, Speech and Dialogue*, Lecture Notes in Computer Science. Springer-Verlag, Berlin-Heidelberg-New York, 2002.
4. Jorge Graña, Fco. Mario Barcala, and Jesús Vilares. Formal methods of tokenization for part-of-speech tagging. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 240–249. Springer-Verlag, Berlin-Heidelberg-New York, 2002.
5. Jorge Graña, Jean-Cédric Chappelier, and Manuel Vilares. Integrating external dictionaries into stochastic part-of-speech taggers. In Galia Angelova, Kalina Bontcheva, Ruslan Mitkov, Nicolas Nocolov and Nokolai Nikolov, editors, *Euro-Conference Recent Advances in Natural Language Processing. Proceedings*, pages 122–128, Tzigov Chark, Bulgaria, 2001.
6. Christian Jacquemin and Evelyne Tzoukermann. NLP for term variant extraction: synergy between morphology, lexicon and syntax. In Tomek Strzalkowski, editor, *Natural Language Information Retrieval*, volume 7 of *Text, Speech and Language Technology*, pages 25–74. Kluwer Academic Publishers, Dordrecht/Boston/London, 1999.
7. Frederick Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, MA, 1998.
8. Mervyn F. Lang. *Spanish Word Formation: Productive Derivational Morphology in the Modern Lexis.* Croom Helm. Routledge, London and New York, 1990.
9. Christer Samuelsson. Morphological tagging based entirely on bayesian inference. In Robert Eklund, editor, *Proceedings of the 9th Nordic Conference on Computational Linguistics*, Stockholm, Sweden, 1993.
10. Jesús Vilares, Fco. Mario Barcala, and Miguel A. Alonso. Using syntactic dependency-pairs conflation to improve retrieval performance in Spanish. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science,*, pages 381–390. Springer-Verlag, Berlin-Heidelberg-New York, 2002.
11. Jesús Vilares, David Cabrero, and Miguel A. Alonso. Applying productive derivational morphology to term indexing of Spanish texts. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2004 of *Lecture*

Notes in Computer Science, pages 336–348. Springer-Verlag, Berlin-Heidelberg-New York, 2001.

12. Jesús Vilares, Manuel Vilares, and Miguel A. Alonso. Towards the development of heuristics for automatic query expansion. In Heinrich C. Mayr, Jiri Lazansky, Gerald Quirchmayr, and Pavel Vogel, editors, *Database and Expert Systems Applications*, volume 2113 of *Lecture Notes in Computer Science*, pages 887–896. Springer-Verlag, Berlin-Heidelberg-New York, 2001.