

Automatic term categorization by extracting knowledge from the Web

Leonardo Rigutini, Ernesto Di Iorio, Marco Ernandes, Marco Maggini

Dipartimento di Ingegneria dell'Informazione

Università di Siena

Via Roma 56, I-53100 - Siena - Italy

{rigutini, diorio, ernandes, maggini}@dii.unisi.it

Abstract. This paper addresses the problem of categorizing terms or lexical entities into a predefined set of semantic domains exploiting the knowledge available on-line in the Web. The proposed system can be effectively used for the automatic expansion of thesauri, limiting the human effort to the preparation of a small training set of tagged entities. The classification of terms is performed by modeling the contexts in which terms from the same class usually appear. The Web is exploited as a significant repository of contexts that are extracted by querying one or more search engines. In particular, it is shown how the required knowledge can be obtained directly from the snippets returned by the search engines without the overhead of document downloads. Since the Web is continuously updated "World Wide", this approach allows us to face the problem of open-domain term categorization handling both the geographical and temporal variability of term semantics. The performances attained by different text classifiers are compared, showing that the accuracy results are very good independently of the specific model, thus validating the idea of using term contexts extracted from search engine snippets. Moreover, the experimental results indicate that only very few training examples are needed to reach the best performance (over 90% for the F1 measure).

1 Introduction

Term categorization is a key task in the Text Mining research area. In fact, the availability of complete and up-to-date lexical knowledge bases is becoming a central issue for automatic text processing applications, especially when dealing with rich and fast changing document collections like the Web. The maintenance of thesauri, gazetteers, and domain-specific lexicons usually requires a large amount of human effort to track changes and new additions of lexical entities. The expansion of domain-specific lexicons consists in adding a set of new and unknown terms to a predefined set of domains. In other words, a lexicon or even a more articulated structure, like an ontology [3], can be populated by associating each unknown lexical entity to one or more specific categories. Thus, the goal of term categorization is to label a lexical entity using a set of semantic themes (i.e. disciplines, domains). Domain-specific lexicons have been used in *word-sense disambiguation* [8], *query-expansion* and *cross-lingual text categorization* [10]. Several proposals to face the problem of the *automatic expansion of ontologies and thesauri* are proposed in the literature [5, 11]. In [2], the *exogenous* and the *endogenous* categorization approaches are proposed for training an

automatic term classifier. The exogenous classification of lexical entities is inspired by corpus-based techniques for Word Sense Disambiguation [12]. The idea is that the sense of a term can be inferred by the context in which it appears. On the other side, the endogenous classification relies only on the statistical information embedded within the sequence of characters that constitute the lexical entity itself. In [1], the authors approach this problem as the dual of text categorization. They use a set of unlabeled documents to learn associations between terms and domains and then to represent each term in the space of these documents. However, the use of a predefined collection of documents to extract knowledge can be quite limitative in high dynamical and rich environments, like the Web, where the lexical entities involved are extremely variable, since different languages, cultures, geographical regions, domains and times of writing may coexist.

In this paper we propose an approach to term categorization that exploits the observation that the Web is a complete knowledge base that is updated continuously and is available on-line through search engines. Many recent attempts to extract the information embedded in Web documents for human-level knowledge handling are reported in the literature, such as [4] in the field of Web-based Question Answering. Nevertheless, as far as we concern, none of these systems is oriented to term categorization. Search engines as GoogleTM answer user queries returning a list of Web links along with a brief excerpt of the documents directly related to the queries. These passages, called *snippets*, represent a condensed and query-relevant version of the document contents and are designed to convey sufficient information to guide the selection of the appropriate results. Thus, snippets can provide a relevant textual context related to the lexical entities used in the query: the words in the snippets can be used to obtain a set of features for representing the query term and this representation can be used to train a classifier. The experimental results show that the information provided by the snippets is effective to attain very good classification performances, thus avoiding the overhead due to the downloads of the referred documents. Interestingly, the choice of appropriate classifier models and feature selection schemes allows us to use only few examples per class in the learning phase. Hence, the Web and, in particular, search engines can be profitably used as sources of textual corpora to automatically generate domain-specific and language-independent thesauri. This approach is general and admits any granularity in the classification. In addition, the self-updating nature of the Web can help to face the problem of thesaurus maintenance in fast evolving environments.

The paper is structured as follows. In the next section the architecture of the system is described in details, explaining the different models used in the experiments. Section 3 reports the configuration and the results of the experiments that were aimed at assessing the system performance and at comparing the possible design choices. Finally, in section 4 the conclusions and the directions for future research are presented.

2 System description

The system for term categorization is composed by two main modules as sketched in Figure 1. The *training module* is used to train the classifier from a set of labeled examples, whereas the *entity classification module* is applied to predict the appropriate category for a given input entity. Both modules exploit the Web to obtain

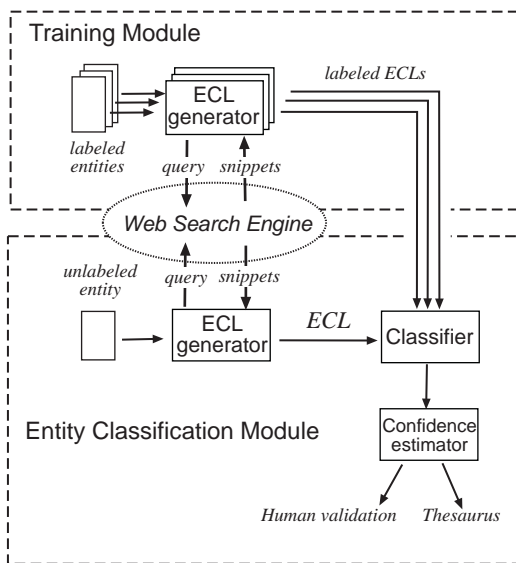


Figure 1. Block diagram of the term categorization system.

an enriched representation of each entity. Basically the entities are transformed into queries and then the snippets obtained by one or more search engines are analyzed to build the *Entity Context Lexicon* ECL_e of each entity e :

$$e \xrightarrow{\text{Search Engine}} ECL_e.$$

In the training step, a set of labeled entities is used to train an automatic classifier to assign an ECL to one category out of a pre-defined set of classes $C = C_1, C_2, \dots, C_k$. For each labeled entity e_l , its ECL_l is computed and provided to the classifier as a training example.

To classify an unknown entity e_u , the corresponding ECL_u is obtained by querying the selected Web search engines. Then, the classifier is used to assign the resulting ECL_u to one class of the category set C . The confidence of the classifier output is evaluated to decide if the category assignment is reliable or if a human judgment is required. In the latter case, the human feedback can be used to expand the training set.

2.1 The ECL generator

This module builds the *Entity Context Lexicon* for a given entity e by analyzing the set of snippets $SN_e = \{snip_1(e), \dots, snip_S(e)\}$ obtained by issuing a query Q_e to one or more search engines. In the current implementation Q_e contains only the terms that compose the entity (f.i. Q_e is the string “New York”). Query expansion techniques will be evaluated in the future developments of the system. The number S of snippets is a system parameter and allows us to include a different number of independent sources of entity contexts. Usually we would like to keep S as small as possible in order to include only the top-ranked results, assuming that the ranking algorithm exploited by the search engines is enough reliable to provide the most relevant and significant results in the top positions of the ranking list. By exploiting more search engines we can enforce this property since the S snippets can be obtained by exploring each result list up to a lower level. However, this advantage is achieved at the cost of a more complicated preprocessing to eliminate duplicate results. Hence, in the current implementation of the system we decided to exploit only one search engine (i.e. Google).

The ECL_e of a given entity e is simply the set of the context terms extracted from the snippets SN_e . For each word $w_k \in ECL_e$, the following statistics are stored:

- the *word count* $wc_{k,e}$ counts the occurrences of w_k in SN_e , i.e. $wc_{k,e} = \#\{w_k \in SN_e\}$;
- the *snippet count* $sc_{k,e}$ is the number of snippets in SN_e containing the word w_k , i.e. $sc_{k,e} = \#\{snip(e) \in SN_e | w_k \in snip(e)\}$.

In order to avoid the inclusion of not significant terms, we can filter the set of the selected terms by means of a *stop-words* list. This list must be properly constructed in order to consider all the languages that are managed by the system.

2.2 The classifier module

In the system each entity e is characterized by the corresponding set of context terms, ECL_e . Hence, the term categorization task can be viewed as a text classification problem where a feature vector is associated to the ECL_e to be classified. Different term weighting schemes and classifier models can be exploited in this module. In this work, several models commonly used in text classification tasks have been tested: *Support Vector Machine (SVM)*, *Naive Bayes (NB)* and *Complement Naive Bayes (CNB)*. Moreover, we propose a prototype-based model particularly suited for this task called *Class-Context-Lexicon (CCL) classifier*, that performs the classification of each entity by evaluating the similarity between the entity and the prototype lexicons.

Support Vector Machine (SVM). The SVM model [6] assumes to use a mapping function Φ that transforms the input vectors into points of a new space, usually characterized by a higher number of dimensions than the original one. In this new space, the learning algorithm estimates the optimal hyperplane that separates the positive from the negative training examples for each class. The model does not require an explicit definition of the mapping function Φ , but exploits a *kernel function* $K(x_1, x_2)$ that computes the scalar product of the images of the points x_1 and x_2 , i.e. $K(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle$. The choice of an appropriate kernel function is the basic design issue of the SVM model. The training algorithm selects a subset of the training examples, the *support vectors*, that define the

separation hyperplane with the maximum margin between the class and its complement.

A different SVM is trained for each class C_j using the ECL_e of the entities labeled with class C_j as positive examples and a subset of the other training entities as negative examples. Each ECL_e is mapped to a feature vector by adopting a specific term weighting function as described in the following. When an entity ECL_u has to be categorized, each model returns the distance between ECL_u and its separation hyperplane. This value ranges in $[-1, 1]$ and can be considered as the similarity score between ECL_u and the class represented by the model.

Naive Bayes. The Naive Bayes classifier estimates the posterior probability of a category given the ECL . Using the Bayes rule, this value can be evaluated by estimating the likelihood of the ECL given the class:

$$P(C_j|ECL_i) = \frac{P(ECL_i|C_j)P(C_j)}{P(ECL_i)}.$$

$P(ECL_i)$ is a normalization factor constant for all categories and it can be ignored. Considering the words in the ECL as independent events, the likelihood of an unknown ECL_u can be evaluated as:

$$P(ECL_u|C_j) = \prod_{w_k} P(w_k|C_j)^{\#\{w_k \in ECL_u\}}.$$

In training phase, each $P(w_k|C_j)$ can be estimated from the frequency of the term w_k in the class C_j .

Complement Naive Bayes. This classifier [9] estimates the posterior probability as $P(C_j|ECL_i) = 1 - P(\overline{C_j}|ECL_i)$, where \overline{C} indicates the complement of C . In this way, the probability $P(\overline{C_j}|ECL_i)$ can be easily estimated similarly as in the Naive Bayes model:

$$P(\overline{C_j}|ECL_i) = \frac{P(ECL_i|\overline{C_j})P(\overline{C_j})}{P(ECL_i)}.$$

Each $P(w_k|\overline{C_j})$ is approximated by the frequency of the term w_k in the complement class $\overline{C_j}$. This approach is particularly suited when only few labeled examples are available for each category C_j .

CCL classifier. Following the idea that similar entities appear in similar contexts, we exploited a new type of profile-based classifier. An independent classifier is trained to model each class. The profile of a given class is obtained by merging the $ECLs$ of the training entities associated to that class. Each term in the profile is associated to a weight evaluated using a given weighting function W . The profile represents the lexicon generated by all the training entities for the j^{th} class C_j and we indicate it as *Class Context Lexicon (CCL)*. The CCL is simply the set of the context terms extracted from the $ECLs$ associated to the entities labeled with the corresponding class. Similarly to the case of the $ECLs$, for each word $w_k \in CCL_j$ the following statistics are stored:

- the *word count* $w_{c_{k,j}}$ counts the occurrences of w_k in the class C_j , i.e. $w_{c_{k,j}} = \#\{w_k \in CCL_j\}$. It is the sum of the $w_{c_{k,e}}$ of the $ECLs$ used to build the CCL ;
- the *snippet count* $sc_{k,j}$ is the number of snippets in class C_j containing the word w_k , i.e. $sc_{k,j} = \#\{snip \in CCL_j | w_k \in snip\}$. It is the sum of the $sc_{k,e}$ of the $ECLs$ used to build the CCL .

When an unlabeled ECL_u is to be classified, each classifier returns a score indicating the membership degree of the ECL_u with respect to the corresponding class. First, the weights of the terms in ECL_u are evaluated using the weighting function W and then the similarity between ECL_u and each CCL_j is computed using a given similarity function.

The CCL model (as also the SVM classifier) exploits a function $W_{ECL}[w]$ that assigns a weight to each component w of an ECL (or CCL). In the experiments, we tested the most commonly used weighting schemes: *binary*, *term frequency (tf)* and *term frequency-inverse document frequency (tfidf)*. Moreover, we defined another weighting function, called *snippet frequency-inverse class frequency (sfcf)*, that combines the frequency in the snippets of each term with its distribution in each class. Being L a lexicon (ECL or CCL), $sc_{k,L}$ the snippet count of the term w_k in L and sc_L the total number of snippets associated to L , the weighting function is defined as:

$$sfcf(w_k, L) = \left(\frac{sc_{w,L}}{sc_L} \right) \cdot \frac{1}{\#\{C | w_k \in C\}}.$$

When evaluating weights of an ECL , $sc_{w,L} = sc_{w,ECL}$ whereas in weighting terms of a CCL , $sc_{w,L} = sc_{w,CCL}$. As reported in section 3, this weighting scheme yields better performances especially when used with the CCL classifier.

Finally, the CCL classifier requires the definition of a similarity function. This function is used to compare an ECL with a CCL and yields high values when the two sets are considered similar. The most commonly used functions in automatic text processing applications are the *Euclidean similarity* and *Cosine similarity*. We also introduced a new similarity function called *Gravity*. These functions are defined as follows.

- *Euclidean similarity function.* It derives from the Euclidean distance function for vectorial spaces.

$$E(ECL_e, CCL_j) = \frac{1}{\|ECL_e - CCL_j\|}.$$

- *Cosine similarity function.* It measures the cosine of the angle formed by the two vectors.

$$C(ECL_e, CCL_j) = \frac{\langle ECL_e, CCL_j \rangle}{\|ECL_e\| \cdot \|CCL_j\|}.$$

- *Gravity similarity function.* It combines the Euclidean function, that is sensitive to the terms not shared by the two vectors, and the cosine correlation, that mainly depends on the shared terms. This function performs better when the number of terms is small (i.e. when the training set used to build the $CCLs$ contains few entities).

$$G(ECL_e, CCL_j) = \frac{\langle ECL_e, CCL_j \rangle}{\|ECL_e - CCL_j\|^2}.$$

In the previous expressions, the euclidean distance and the cosine values are computed as

$$\|ECL_e - CCL_j\| = \sqrt{\sum_w (W_{ECL_e}[w] - W_{CCL_j}[w])^2}$$

and

$$\langle ECL_e, CCL_j \rangle = \sum_w W_{ECL_e}[w] \cdot W_{CCL_j}[w],$$

where $W_{ECL_e}[w]$ and $W_{CCL_j}[w]$ are the weights of the term w in ECL_e and CCL_j , respectively.

The CCL classifier, being a prototype-based model, guarantees the modularity of the classifier when adding new categories in the domain set. In fact, each class is modeled by a CCL that is built independently of the examples in the other classes. When a new category is added, it is simply required to learn the CCL of the new class without modifying the previously computed CCL s. Also to the Naive Bayes classifier guarantees this kind of modularity, but the experimental results showed that the CCL classifier is able to attain better performances. All the others models, instead, use negative examples in the learning phase and, therefore, they require a new training step for all the categories when a new domain is added.

3 Experimental results

We selected 8 categories (*soccer, music, location, computer, politics, food, philosophy, medicine*) and for each of them we searched for predefined gazetteers on the Web. Then, we randomly sampled these lists of terms in order to collect 200 entities for each class¹. The *soccer, music* and *politics* classes contain many proper names of players, coaches, teams, singers, musicians, bands, politicians, and political parties. The *location* category collects names of cities and countries, while the *computer* category mainly lists brands of computer devices and software. The *food* class contains names of dishes and typical foods, whereas in the *philosophy* category, there are the terms for various philosophical currents and concepts (e.g. casualism, illuminism and existentialism). Finally, in the *medicine* class, there are terms related to pathologies and treatments (e.g. dermatitis).

The dataset was split into two subsets, both composed by 100 randomly sampled entities: a learning collection and a test set. For each experiment we randomly partitioned the learning collection into 5 distinct training sets in order to average the results with a five-fold-validation.

The precision and recall measures were used to evaluate the system performance. Given a class C_j , the corresponding precision and recall values are defined as

$$Pr_j = \frac{TP_j}{TP_j + FP_j} \quad Re_j = \frac{TP_j}{TP_j + FN_j},$$

where TP_j is the number of the examples correctly assigned to C_j (the true positives), FP_j is the number of wrong assignments to C_j (the false positives), and FN_j is the number of examples incorrectly not assigned to C_j (the false negatives). Since the test sets for each class are balanced, we averaged these values over the set of classes (i.e. we report the *Micro Average*). Precision and recall were combined using the classical F_1 value ($F_1 = \frac{2 \cdot Pr \cdot Re}{Pr + Re}$).

We performed a preliminary set of experiments to determine the optimal number S of snippets to be used in the construction of the ECL s. We found that there is no significant improvement for $S > 10$, thus we decided to perform the following tests using $S = 10$.

Using the entities in the learning collection we extracted different training sets with an increasing number of entities. We indicate as LS_M the training set containing M entities per category (i.e. a total of $8M$ entities).

The first test aimed at comparing the performances of the CCL -based classifier using the different weighting and similarity functions and at evaluating the influence of the size of the training set. Figure

2 reports the plots of the F_1 value for the three best configurations (we decided to report only these cases in order to improve the plot readability). The best performing model is the one that exploits the *Gravity similarity function* and the *sfcf* weighting scheme. In fact, the gravity similarity shows a slightly better behavior for small learning sets. Using this configuration for the CCL classifier, we obtain about 90% for the F_1 value using just 5–10 examples per class. No evident improvement is achieved by adding more examples, and this result is probably due to the presence of some intrinsically ambiguous entities in the dataset. However, the system performance is very satisfactory and validates the effectiveness of the proposed approach. Table 1 collects the words with the highest score in the profile of each

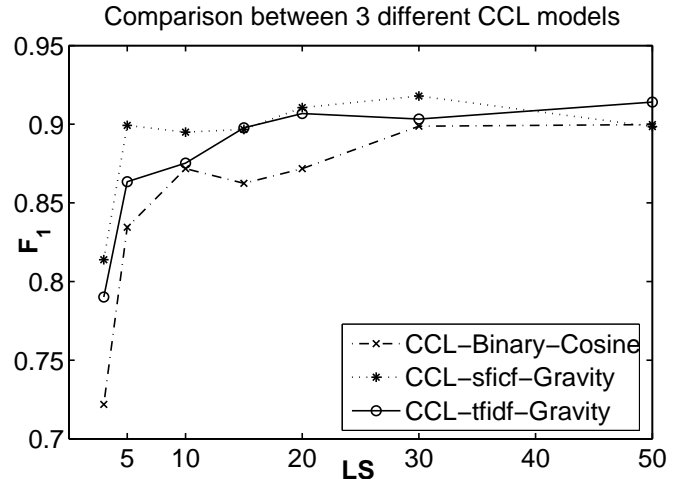


Figure 2. Plot of the F_1 values with respect to the training set size for different weighting and similarity functions in the CCL classifier. Each value is the average on 5 different runs exploiting different training sets.

class using the best classifier configuration, CCL^* . It can be noticed that words in different languages (English and Italian) appear in the same set, thus showing that the system can naturally adapt its behavior to a multi-lingual context.

Soccer:	league, soccer, goal, lega, campionato, rigore, attaccante, calciomercato
Politics:	partito, republican, governors, comunista, adams, hoover, clinton, coolidge
Location:	islands, india, geography, country, map, flag, tourism, city
Medicine:	atrial, infections, cavernous, hemangioma, microscopy, vascular, electron, genetic
Food:	pumpkin, bread, oil, sirloin, wheat, chicken, honey, steak
Organization:	visio, micro, silicon, drivers, laptop, virus, batteries, software
Music:	lyrics, guitar, willie, albums, chords, music, smith, fm
Philosophy:	searches, empiricus, definition, outlines, theory, knowledge, philosophical, doctrine

Table 1. The top scored words for each CCL in the best performing classifier.

After we individuated the best configuration for the CCL classifier, we compared it with the *Naive Bayes*, *Complement Naive Bayes*

¹ The dataset is available at <http://airgroup.dii.unisi.it/dataset/WCD.tar.gz>

and *Support Vector Machine*. We used *SVM-light*² as implementation of the SVM classifier [7] and we exploited the linear kernel that has been shown to be the best performing one in text classification tasks. In the figure 3, we notice that the *CNB* classifier attains the best performances, although they are very similar to those of the *CCL*-based classifier. Moreover, the SVM classifier shows worse performances than the other models. This model, in fact, requires a great number of support vectors to locate the best separation hyperplane. In this application, the size of training sets is quite small thus reducing the effectiveness of this model. In fact, from figure 3 we can notice that the SVM model obtains significant performance improvements when increasing of the learning set size.

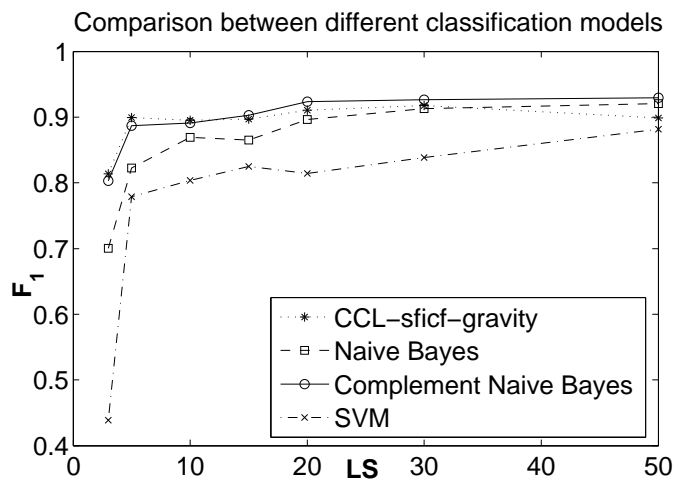


Figure 3. Plot of the F_1 values with respect to the training set size (M entities per class) for the different classifier models. Each value is the average on 5 different runs exploiting different training sets.

Globally, the performance saturates when increasing the learning set size, showing that the system is able to attain good precision and recall values even with a quite limited set of examples. In particular, we individuated $M = 20$ as the best compromise between performances and learning set cardinality, being no significant increase in the performance for larger sets.

4 Conclusions and future work

In this work we proposed a system for Web based term categorization, oriented to automatic thesaurus construction. The idea is that “terms from a same semantic category” should appear in very similar contexts, i.e. that contain approximately the same words. Based on this assumption, the system builds an *Entity Context Lexicon (ECL)* for each entity, that is the vocabulary composed by the words extracted from the first 10 snippets returned by Google submitting e as a query. Each *ECL* can be considered as the representation of the *term* in a feature space and an automatic classifier can be trained to categorize it into one category out of a predefined set of classes. In this work, we compared some popular classification models and we also proposed a new profile-based model that exploits the concept of class lexicon (*Context Class Lexicon, CCL*). The new model performs well especially when a small number of training examples is provided and it does not require a global retraining step if a new

domain is added to the category set. Even if the Complement Naive Bayes model (*CNB*) yields the best performances, it does not have these two positive features. The experiments proved that with just 20 training examples per class, the system averages with over 90% of the F_1 measure. These results are very promising and we expect a further improvement after a tuning of the system design. Additional tests have been planned considering a multi-label classification of each entity and to verify the robustness of the system in “out of topic” cases, i.e. when the correct category is not present in the taxonomy.

REFERENCES

- [1] Henri Avancini, Alberto Lavelli, Bernardo Magnini, Fabrizio Sebastiani, and Roberto Zanolì, ‘Expanding domain-specific lexicons by term categorization’, in *SAC ’03: Proceedings of the 2003 ACM symposium on Applied computing*, pp. 793–797, New York, NY, USA, (2003). ACM Press.
- [2] F. Cerbah, ‘Exogenous and endogenous approaches to semantic categorization of unknown technical terms’, in *In Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pp. 145–151, Saarbrücken, Germany, (2000).
- [3] M. Ciaramita and M. Johnson, ‘Supersense tagging of unknown nouns in wordnet.’, in *In Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sapporo, Japan, (2003).
- [4] M. Ernandes, G. Angelini, and M. Gori, ‘WebCrow: A web-based system for crossword solving’, in *In Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, Pittsburgh, PA, (2005).
- [5] M.A. Hearst, ‘Automatic acquisition of hyponyms from large text corpora’, in *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pp. 539–545, Nantes, France, (1992).
- [6] T. Joachims, ‘Text categorization with support vector machines: Learning with many relevant features.’, in *Proceedings of ECML ’98*, (1998).
- [7] T. Joachims, ‘Estimating the generalization performance of a svm efficiently’, in *Proceedings of the International Conference on Machine Learning*, (2000).
- [8] B. Magnini, C. Strapparava, G. Pezzulo, and A. Gliozzo, ‘The role of domain information in word sense disambiguation’, *Natural Language Engineering*, 8(4), 359–373, (2002).
- [9] J.D. Rennie, L. Shih, J. Teevan, and D. Karger, ‘Tackling the poor assumptions of naive bayes text classifiers’, in *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington DC, (2003).
- [10] L. Rigutini, B. Liu, and M. Maggini, ‘An em based training algorithm for cross-language text categorization’, in *In Proceedings of the Web Intelligence Conference (WI)*, pp. 529–535, Compiègne, France, (2005).
- [11] N. Uramoto, ‘Positioning unknown words in a thesaurus by using information extracted from a corpus.’, in *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pp. 956–961, Copenhagen, (1996).
- [12] D. Yarowski, ‘Word sense disambiguation using statistical models of roget’s categories trained on large corpora’, in *In Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, Nantes, France, (1992).

² Available at <http://svmlight.joachims.org/>