SENSEVAL-3: Third International Workshop on the Evaluation of Systems
for the Semantic Analysis of Text, Barcelona, Spain, July 2004
Association for Computational Linguistics

Pattern Abstraction and Term Similarity for Word Sense Disambiguation: IRST at Senseval-3

Carlo Strapparava and **Alfio Gliozzo** and **Claudio Giuliano**
ITC-irst, Istituto per la Ricerca Scientifica e Tecnologica, I-38050 Trento, ITALY
{strappa, gliozzo, giuliano}@itc.it

Abstract

This paper summarizes IRST's participation in Senseval-3. We participated both in the English all-words task and in some lexical sample tasks (English, Basque, Catalan, Italian, Spanish). We followed two perspectives. On one hand, for the all-words task, we tried to refine the Domain Driven Disambiguation that we presented at Senseval-2. The refinements consist of both exploiting a new technique (Domain Relevance Estimation) for domain detection in texts, and experimenting with the use of Latent Semantic Analysis to avoid reliance on manually annotated domain resources (e.g. WORDNET DOMAINS). On the other hand, for the lexical sample tasks, we explored the direction of pattern abstraction and we demonstrated the feasibility of leveraging external knowledge using kernel methods.

1 Introduction

The starting point for our research in the Word Sense Disambiguation (WSD) area was to explore the use of semantic domains in order to solve lexical ambiguity. At the Senseval-2 competition we proposed a new approach to WSD, namely Domain Driven Disambiguation (DDD). This approach consists of comparing the estimated domain of the context of the word to be disambiguated with the domains of its senses, exploiting the property of domains to be features of both texts and words. The domains of the word senses can be either inferred from the learning data or derived from the information in WORDNET DOMAINS.

For Senseval-3, we refined the DDD methodology with a fully unsupervised technique - Domain Relevance Estimation (DRE) - for domain detection in texts. DRE is performed by an expectation maximization algorithm for the gaussian mixture model, which is exploited to differentiate relevant domain information in texts from noise. This refined DDD system was presented in the English all-words task.

Originally DDD was developed to assess the usefulness of domain information for WSD. Thus it did not exploit other knowledge sources commonly used for disambiguation (e.g. syntactic patterns or collocations). As a consequence the performance of the DDD system is quite good for precision (it disambiguates well the "domain" words), but as far as recall is concerned it is not competitive compared with other state of the art techniques. On the other hand DDD outperforms the state of the art for unsupervised systems, demonstrating the usefulness of domain information for WSD.

In addition, the DDD approach requires domain annotations for word senses (for the experiments we used WORDNET DOMAINS, a lexical resource developed at IRST). Like all manual annotations, such an operation is costly (more than two man years have been spent for labeling the whole WORDNET DOMAINS structure) and affected by subjectivity. Thus, one drawback of the DDD methodology was a lack of portability among languages and among different sense repositories (unless we have synset-aligned WordNets).

Besides the improved DDD, our other proposals for Senseval-3 constitute an attempt to overcome these previous issues.

To deal with the problem of having a domain-annotated WORDNET, we experimented with a novel methodology to automatically acquire domain information from corpora. For this aim we estimated term similarity from a large scale corpus, exploiting the assumption that semantic domains are sets of very closely related terms. In particular we implemented a variation of Latent Semantic Analysis (LSA) in order to obtain a vector representation for words, texts and synsets. LSA performs a dimensionality reduction in the feature space describing both texts and words, capturing implicitly the notion of semantic domains required by DDD. In order to perform disambiguation, LSA vectors have been estimated for the synsets in WORDNET. We participated in the English all-words task also with a first prototype (DDD-LSA) that exploits LSA instead of WORDNET DOMAINS.

Task	*Systems*	
English All-Words	DDD	DDD-LSA
English Lex-sample	Kernels-WSD	Ties
Italian Lex-Sample	Kernels-WSD	Ties
Basque Lex-Sample	Kernels-WSD	
Catalan Lex-Sample	Kernels-WSD	
Spanish Lex-Sample	Kernels-WSD	

Table 1: IRST participation at Senseval-3

As far as lexical sample tasks are concerned, we participated in the English, Italian, Spanish, Catalan, and Basque tasks. For these tasks, we explored the direction of pattern abstraction for WSD. Pattern abstraction is an effective methodology for WSD (Mihalcea, 2002). Our preliminary experiments have been performed using *TIES*, a generalized Information Extraction environment developed at IRST that implements the boosted wrapper induction algorithm (Freitag and Kushmerick, 2000). The main limitation of such an approach is, once more, the integration of different knowledge sources. In particular, paradigmatic information seems hard to be represented in the TIES framework, motivating our decision to exploit kernel methods for WSD.

Kernel methods is an area of recent interest in Machine Learning. Kernels are similarity functions between instances that allows to integrate different knowledge sources and to model explicitly linguistic insights inside the powerful framework of support vector machine classification. For Senseval-3 we implemented the *Kernels-WSD* system, which exploits kernel methods to perform the following operations: (i) pattern abstraction; (ii) combination of different knowledge sources, in particular domain information and syntagmatic information; (iii) integration of unsupervised term proximity estimation in the supervised framework.

The paper is structured as follows. Section 2 introduces LSA and its relations with semantic domains. Section 3 presents the systems for the English all-words task (i.e. DDD and DDD-LSA). In section 4 our supervised approaches are reported. In particular the TIES system is described in section 4.1, while the approach based on kernel methods is discussed in section 4.2.

2 Semantic Domains and LSA

Domains are common areas of human discussion, such as economics, politics, law, science etc., which are at the basis of lexical coherence. A substantial part of the lexicon is composed by "domain words", that refer to concepts belonging to specific domains. In (Magnini et al., 2002) it has been claimed that domain information provides generalized features at the paradigmatic level that are useful to discriminate among word senses.

The WORDNET DOMAINS[1] lexical resource is an extension of WORDNET which provides such domain labels for all synsets (Magnini and Cavaglià, 2000). About 200 domain labels were selected from a number of dictionaries and then structured in a taxonomy according to the Dewey Decimal Classification (DDC). The annotation methodology was mainly manual and took about 2 person years.

WORDNET DOMAINS has been proven a useful resource for WSD. However some aspects induced us to explore further developments. These issues are: (i) it is difficult to find an objective a-priori model for domains; (ii) the annotation procedure followed to develop WORDNET DOMAINS is very expensive, making hard the replicability of the lexical resource for other languages or domain specific sub-languages; (iii) the domain distinctions are rigid in WORDNET DOMAINS, while a more "fuzzy" association between domains and concepts is often more appropriate to describe term similarity.

In order to generalize the domain approach and to overcome these issues, we explored the direction of unsupervised learning on a large-scale corpus (we used the BNC corpus for all the experiments described in this paper).

In particular, we followed the LSA approach (Deerwester et al., 1990). In LSA, term co-occurrences in the documents of the corpus are captured by means of a dimensionality reduction operated on the term-by-document matrix. The resulting LSA vectors can be exploited to estimate both term and document similarity. Regarding document similarity, Latent Semantic Indexing (LSI) is a technique that allows one to represent a document by a LSA vector. In particular, we used a variation of the *pseudo-document* methodology described in (Berry, 1992). Each document can be represented in the LSA space by summing up the normalized LSA vectors of all the terms contained in it.

By exploiting LSA vectors for terms, it is possible to estimate domain vectors for the synsets of WORDNET, in order to obtain similarity values between concepts that can be used for synset clustering and WSD. Thus, term and document vectors can be used instead of WORDNET DOMAINS for WSD and other applications in which term similarity and domain relevance estimation is required.

[1] WORDNET DOMAINS is freely available for research purposes at `wndomains.itc.it`

3 All-Words systems: DDD and DDD-LSA

DDD with DRE. DDD assignes the right sense of a word in its context comparing the domain of the context to the domain of each sense of the word. This methodology exploits WORDNET DOMAINS information to estimate both the domain of the textual context and the domain of the senses of the word to disambiguate.

The basic idea to estimate domain relevance for texts is to exploit lexical coherence inside texts. A simple heuristic approach to this problem, used in Senseval-2, is counting the occurrences of domain words for every domain inside the text: the higher the percentage of domain words for a certain domain, the more relevant the domain will be for the text.

Unfortunately, the simple local frequency count is not a good domain relevance measure for several reasons. Indeed irrelevant senses of ambiguous words contribute to augment the final score of irrelevant domains, introducing noise. Moreover, the level of noise is different for different domains because of their different sizes and possible differences in the ambiguity level of their vocabularies. We refined the original Senseval-2 DDD system with the Domain Relevance Estimation (DRE) technique. Given a certain domain, DRE distinguishes between relevant and non-relevant texts by means of a Gaussian Mixture model that describes the frequency distribution of domain words inside a large-scale corpus (in particular we used the BNC corpus also in this case). Then, an Expectation Maximization algorithm computes the parameters that maximize the likelihood of the model on the empirical data (Gliozzo et al., 2004).

In order to represent domain information we introduced the notion of Domain Vectors (DV), which are data structures that collect domain information. These vectors are defined in a multidimensional space, in which each domain represents a dimension of the space. We distinguish between two kinds of DVs: (i) *synset vectors*, which represent the relevance of a synset with respect to each considered domain and (ii) *text vectors*, which represent the relevance of a portion of text with respect to each domain in the considered set. The core of the DDD algorithm is based on scoring the comparison of these kinds of vectors. The synset vectors are built considering WORDNET DOMAINS, while in the calculation of scoring the system takes into account synset probabilities on SemCor. The system makes use of a threshold *th-cut*, ranging in the interval [0,1], that allows us to tune the tradeoff between precision and recall.

th-cut	*Prec*	*Recall*	*Attempted*
0.0	0.583	0.583	99.76
0.9	0.729	0.441	60.51

Table 2: DDD on the English all-words task.

Latent Semantic Domains for DDD. As seen in Section 2, it is possible to implement a DDD version that does not use WORDNET DOMAINS and instead it exploits LSA term and document vectors for estimating synset vectors and text vectors, leaving the core of DDD algorithm unchanged. As for text vectors, we used the psedo-document technique also for building synset vectors: in this case we consider the synonymous terms contained in the synset itself.

The system presented at Senseval-3 does not make use of any statistics on SemCor, and consequently it can be considered fully unsupervised. Results are reported in table 3 and do not differ much from the results obtained by DDD in the same task.

th-cut	*Prec*	*Recall*	*Attempted*
0.5	0.661	0.496	75.01

Table 3: DDD-LSA on the English all-words task.

4 Lexical Sample Systems: Pattern abstraction and Kernel Methods

One of the most discriminative features for lexical disambiguation is the lexical/syntactic pattern in which the word appears. A well known issue in the WSD area is the *one sense per collocation* claim (Yarowsky, 1993) stating that the word meanings are strongly associated with the particular collocation in which the word is located. Collocations are sequences of words in the context of the word to disambiguate, and can be associated to word senses performing supervised learning.

Another important knowledge source for WSD is the shallow-syntactic pattern in which a word appears. Syntactic patterns, like lexical patterns, can be obtained by exploiting pattern abstraction techniques on POS sequences. In the WSD literature both lexical and syntactic patterns have been used as features in a supervised learning schema by representing each instance using bigrams and trigrams in the surrounding context of the word to be analyzed[2].

[2]More recently deep-syntactic features have been also considered by several systems, as for example modifiers of nouns and verbs, object and subject of the sentence, etc. In order to

Representing each instance by a "bag of features" presents several disadvantages from the point of view of both machine learning and computational linguistics: (1) Sparseness in the learning data: most of the collocations found in the learning data occur just once, reducing the generalization power of the learning algorithm. In addition most of the collocations found in the test data are often unseen in the training data. (2) Low flexibility for pattern abstraction purposes: bigram and trigram extraction schemata are fixed in advance. (3) Knowledge acquisition bottleneck: the size of the training data is not large enough to cover each possible collocation in the language.

To overcome problems 1 and 2 we investigated some pattern abstraction techniques from the area of Information Extraction (IE) and we adapted them to WSD. To overcome problem 3 we developed Latent Semantic Kernels, which allow us to integrate external knowledge provided by unsupervised term similarity estimation.

4.1 TIES

Our first experiments have been performed exploiting TIES, an environment developed at IRST for IE that induces patterns from the marked entities in the training phase, and then applies those patterns in the test phase in order to assign a category if the pattern is satisfied. For our experiments, we used the Boosted Wrapper Induction (BWI) algorithm (Freitag and Kushmerick, 2000) that is implemented in TIES.

For Senseval-3 we used very few features (lemma and POS). We proposed the system in this configuration as a "baseline" system for pattern abstraction.

Task	*Prec*	*Recall*	*Attempted*
English LS	0.706	0.505	71.50
English LS (coarse)	0.767	0.548	71.50
Italian LS	0.552	0.309	55.92

Table 4: Performance of the TIES system

Our preliminary experiments with BWI have shown that pattern abstraction is very attractive for WSD, allowing us to achieve a very high precision for a restricted number of words, in which the syntagmatic information is sufficient for disambiguation. However, we still had some restrictions. In particular, the integration with different knowledge sources for classification is not trivial.

obtain such features parsing of the data is required. However, we decided to do not use such information, while we plan to introduce it in the next future.

4.2 Kernel-WSD

Our choice of exploiting kernel methods for WSD has been motivated by the observation that pattern-based approaches for disambiguation are complementary to the domain based ones: they require different knowledge sources and different techniques for classification and feature description. Both approaches have to be simultaneously taken into account in order to perform accurate disambiguation. Our aim was to combine them into a common framework.

Kernel methods, e.g. Support Vector Machines (SVMs), are state-of-the-art learning algorithms, and they are successfully adopted in many NLP tasks.

The idea of SVM (Cristianini and Shawe-Taylor, 2000) is to map the set of training data into a higher-dimensional feature space $\mathcal{F}$ via a mapping function $\phi : \aleph \rightarrow \mathcal{F}$, and construct a separating hyperplane with maximum margin (distance between planes and closest points) in the new space. Generally, this yields a nonlinear decision boundary in the input space. Since the feature space is high dimensional, performing the transformation has often a high computational cost. Rather than use the explicit mapping ϕ, we can use a kernel function $K : \aleph \times \aleph \rightarrow \Re$, that corresponds to the inner product in a feature space which is, in general, different from the input space.

Therefore, a kernel function provides a way to compute (efficiently) the separating hyperplane without explicitly carrying out the map ϕ into the feature space - this is called the *kernel trick*. In this way the kernel acts as an interface between the data and the learning algorithm by defining an implicit mapping into the feature space. Intuitively, we can see the kernel as a function that measures the similarity between pairs of objects. The learning algorithm, which compares all pairs of data items, exploits the information encoded in the kernel. An important characteristic of kernels is that they are not limited to vector objects but are applicable to virtually any kind of object representation.

In this work we use kernel methods to combine heterogeneous sources of information that we found relevant for WSD. For each of these aspects it is possible to define kernels independently. Then they are combined by exploiting the property that the sum of two kernels is still a kernel (i.e. $k(x, y) = k_1(x, y) + k_2(x, y)$), taking advantage of each single contribution in an intuitive way[3].

[3]In order to keep the kernel values comparable for different values and to be independent from the length of the examples, we considered the normalized version $\hat{K}(x, y) =$

lsa	*Task*	*Prec*	*Recall*	*Attempted*	*MF-Baseline*
⋆	English LS	0.726	0.726	100	0.552
⋆	English LS (coarse)	0.795	0.795	100	0.645
-	*English LS (no-lsa)*	*0.704*	*0.704*	*100*	*0.552*
-	Basque LS	0.655	0.655	100	0.558
-	Italian LS	0.531	0.531	100	0.183
-	Catalan LS	0.858	0.846	98.62	0.663
-	Spanish LS	0.842	0.842	100	0.677

Table 5: Performance of the Kernels-WSD system

The Word Sense Disambiguation Kernel is defined in this way:

$$K_{WSD}(x,y) = K_S(x,y) + K_P(x,y) \quad (1)$$

where K_S is the Syntagmatic Kernel and K_P is the Paradigmatic Kernel.

The Syntagmatic Kernel. The syntagmatic kernel generalizes the word-sequence kernels defined by (Cancedda et al., 2003) to sequences of lemmata and POSs. Word sequence kernels are based on the following idea: two sequences are similar if they have in common many sequences of words in a given order. The similarity between two examples is assessed by the number (possibly non-contiguous) of the word sequences matching. Non-contiguous occurrences are penalized according to the number of gaps they contain. For example the sequence of words "*I go very quickly to school*" is less similar to "*I go to school*" than "*I go quickly to school*". Different than the bag-of-word approach, word sequence kernels capture the word order and allow gaps between words. The word sequence kernels are parametric with respect to the length of the (sparse) sequences they want to capture.

We have defined the syntagmatic kernel as the sum of n distinct word-sequence kernels for lemmata (i.e. Collocation Kernel - K_C) and sequences of POSs (i.e. POS Kernel - K_{POS}), according to the formula (for our experiments we set n to 2):

$$K_S(x,y) = \sum_{i=1}^{n} K_{C_i}(x,y) + \sum_{i=1}^{n} K_{POS_i}(x,y) \quad (2)$$

In the above definition of syntagmatic kernel, only exact lemma/POS matches contribute to the similarity. One shortcoming of this approach is that (near-)synonyms will never be considered similar. We address this problem by considering soft-matching of words employing a term similarity measure based on LSA[4]. In particular we considered equivalent two words having the same POS and a similarity value higher than an empirical threshold. For example, if we consider as equivalent the terms *Ronaldo* and *football_player* the sequence *The football_player scored the first goal* can be considered equivalent to the sentence *Ronaldo scored the first goal*. The properties of the kernel methods offer a flexible way to plug additional information, in this case unsupervised (we could also take this information from a semantic network such as WORDNET).

The Paradigmatic Kernel. The paradigmatic kernel takes into account the paradigmatic aspect of sense distinction (i.e. domain aspects) (Gliozzo et al., 2004). For example the word *virus* can be disambiguated by recognizing the domain of the context in which it is placed (e.g. *computer_science* vs. *biology*). Usually such an aspect is captured by "bag-of-words", in analogy to the Vector Space Model, widely used in Text Categorization and Information Retrieval. The main limitation of this model for WSD is the knowledge acquisition bottleneck (i.e. the lack of sense tagged data). Bag of words are very sparse data that require a large scale corpus to be learned. To overcome such a limitation, Latent Semantic Indexing (LSI) can provide a solution.

Thus we defined a paradigmatic kernel composed by the sum of a "traditional" bag of words kernel and an LSI kernel (Cristianini et al., 2002) as defined by formula 3:

$$K_P(x,y) = K_{BoW}(x,y) + K_{LSI}(x,y) \quad (3)$$

where K_{BoW} computes the inner product between the vector space model representations and K_{LSI} computes the cosine between the LSI vectors representing the texts.

$K(x,y)/sqrt(K(x,x)K(y,y))$

[4]For languages other than English, we did not exploit this soft-matching and the K_{LSI} kernel described below. See the first column in the table 5.

Table 5 displays the performance of Kernel-WSD. As a comparison, we also report the figures on the English task without using LSA. The last column reports the recall of the most-frequent baseline.

Acknowledgments

Claudio Giuliano is supported by the IST-Dot.Kom project sponsored by the European Commission (Framework V grant IST-2001-34038). TIES and the kernel package have been developed in the context of the Dot.Kom project.

References

M. Berry. 1992. Large-scale sparse singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49.

N. Cancedda, E. Gaussier, C. Goutte, and J.M. Renders. 2003. Word-sequence kernels. *Journal of Machine Learning Research*, 3(6):1059–1082.

N. Cristianini and J. Shawe-Taylor. 2000. *Support Vector Machines*. Cambridge University Press.

N. Cristianini, J. Shawe-Taylor, and H. Lodhi. 2002. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18(2):127–152.

S. Deerwester, S. T. Dumais, G. W. Furnas, T.K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

D. Freitag and N. Kushmerick. 2000. Boosted wrapper induction. In *Proc. of AAAI-00*, pages 577–583, Austin, Texas.

A. Gliozzo, C. Strapparava, and I. Dagan. 2004. Unsupervised and supervised exploitation of semantic domains in lexical disambiguation. *Computer Speech and Language*, Forthcoming.

B. Magnini and G. Cavaglià. 2000. Integrating subject field codes into WordNet. In *Proceedings of LREC-2000*, Athens, Greece, June.

B. Magnini, C. Strapparava, G. Pezzulo, and A. Gliozzo. 2002. The role of domain information in word sense disambiguation. *Natural Language Engineering*, 8(4):359–373.

R. F. Mihalcea. 2002. Word sense disambiguation with pattern learning and automatic feature selection. *Natural Language Engineering*, 8(4):343–358.

D. Yarowsky. 1993. One sense per collocation. In *Proceedings of ARPA Human Language Technology Workshop*, pages 266–271, Princeton.