Syntagmatic Kernels: a Word Sense Disambiguation Case Study

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

Abstract

In this paper we present a family of kernel functions, named Syntagmatic Kernels, which can be used to model syntagmatic relations. Syntagmatic relations hold among words that are typically collocated in a sequential order, and thus they can be acquired by analyzing word sequences. In particular, Syntagmatic Kernels are defined by applying a Word Sequence Kernel to the local contexts of the words to be analyzed. In addition, this approach allows us to define a semi supervised learning schema where external lexical knowledge is plugged into the supervised learning process. Lexical knowledge is acquired from both unlabeled data and hand-made lexical resources, such as WordNet. We evaluated the syntagmatic kernel on two standard Word Sense Disambiguation tasks (i.e. English and Italian lexical-sample tasks of Senseval-3), where the syntagmatic information plays a crucial role. We compared the Syntagmatic Kernel with the standard approach, showing promising improvements in performance.

1 Introduction

In computational linguistics, it is usual to deal with sequences: words are sequences of letters and syntagmatic relations are established by sequences of words. Sequences are analyzed to measure morphological similarity, to detect multiwords, to represent syntagmatic relations, and so on. Hence modeling syntagmatic relations is crucial for a wide variety of NLP tasks, such as Named Entity Recognition (Gliozzo et al., 2005a) and Word Sense Disambiguation (WSD) (Strapparava et al., 2004).

In general, the strategy adopted to model syntagmatic relations is to provide bigrams and trigrams of collocated words as features to describe local contexts (Yarowsky, 1994), and each word is regarded as a different instance to classify. For instance, occurrences of a given class of named entities (such as *names of persons*) can be discriminated in texts by recognizing word patterns in their local contexts. For example the token *Rossi*, whenever is preceded by the token *Prof.*, often represents the name of a person. Another task that can benefit from modeling this kind of relations is WSD. To solve ambiguity it is necessary to analyze syntagmatic relations in the local context of the word to be disambiguated. In this paper we propose a kernel function that can be used to model such relations, the *Syntagmatic Kernel*, and we apply it to two (English and Italian) lexical-sample WSD tasks of the Senseval-3 competition (Mihalcea and Edmonds, 2004).

In a lexical-sample WSD task, training data are provided as a set of texts, in which for each text a given target word is manually annotated with a sense from a predetermined set of possibilities. To model syntagmatic relations, the typical supervised learning framework adopts as features bigrams and trigrams in a local context. The main drawback of this approach is that non contiguous or shifted col-

locations cannot be identified, decreasing the generalization power of the learning algorithm. For example, suppose that the verb *to score* has to be disambiguated into the sentence "Ronaldo *scored* the goal", and that the sense tagged example "the football player `scores#1` the first goal" is provided for training. A traditional feature mapping would extract the bigram $\mathbf{w_{+1}\_w_{+2}}$**:the_goal** to represent the former, and the bigram $\mathbf{w_{+1}\_w_{+2}}$**:the_first** to index the latter. Evidently such features will not match, leading the algorithm to a misclassification.

In the present paper we propose the Syntagmatic Kernel as an attempt to solve this problem. The Syntagmatic Kernel is based on a Gap-Weighted Subsequences Kernel (Shawe-Taylor and Cristianini, 2004). In the spirit of Kernel Methods, this kernel is able to compare sequences directly in the input space, avoiding any explicit feature mapping. To perform this operation, it counts how many times a (non-contiguous) subsequence of symbols u of length n occurs in the input string s, and penalizes non-contiguous occurrences according to the number of the contained gaps. To define our Syntagmatic Kernel, we adapted the generic definition of the Sequence Kernels to the problem of recognizing collocations in local word contexts.

In the above definition of Syntagmatic Kernel, only exact word-matches contribute to the similarity. One shortcoming of this approach is that (near-)synonyms will never be considered similar, leading to a very low generalization power of the learning algorithm, that requires a huge amount of data to converge to an accurate prediction. To solve this problem we provided external lexical knowledge to the supervised learning algorithm, in order to define a "soft-matching" schema for the kernel function. For example, if we consider as equivalent the terms *Ronaldo* and *football_player*, the proposition "*The football_player scored the first goal*" is equivalent to the sentence "*Ronaldo scored the first goal*", providing a strong evidence to disambiguate the latter occurrence of the verb.

We propose two alternative soft-matching criteria exploiting two different knowledge sources: (i) hand made resources and (ii) unsupervised term similarity measures. The first approach performs a soft-matching among all those synonyms words in WordNet, while the second exploits domain relations, acquired from unlabeled data, for the same purpose.

Our experiments, performed on two standard WSD benchmarks, show the superiority of the Syntagmatic Kernel with respect to a classical flat vector representation of bigrams and trigrams.

The paper is structured as follows. Section 2 introduces the Sequence Kernels. In Section 3 the Syntagmatic Kernel is defined. Section 4 explains how soft-matching can be exploited by the Collocation Kernel, describing two alternative criteria: WordNet Synonymy and Domain Proximity. Section 5 gives a brief sketch of the complete WSD system, composed by the combination of different kernels, dealing with syntagmatic and paradigmatic aspects. Section 6 evaluates the Syntagmatic Kernel, and finally Section 7 concludes the paper.

2 Sequence Kernels

The basic idea behind kernel methods is to embed the data into a suitable feature space $\mathcal{F}$ via a mapping function $\phi : \mathcal{X} \rightarrow \mathcal{F}$, and then use a linear algorithm for discovering nonlinear patterns. Instead of using the explicit mapping ϕ, we can use a kernel function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, that corresponds to the inner product in a feature space which is, in general, different from the input space.

Kernel methods allow us to build a modular system, as the kernel function acts as an interface between the data and the learning algorithm. Thus the kernel function becomes the only domain specific module of the system, while the learning algorithm is a general purpose component. Potentially any kernel function can work with any kernel-based algorithm. In our system we use Support Vector Machines (Cristianini and Shawe-Taylor, 2000).

Sequence Kernels (or *String Kernels*) are a family of kernel functions developed to compute the inner product among images of strings in high-dimensional feature space using dynamic programming techniques (Shawe-Taylor and Cristianini, 2004). The Gap-Weighted Subsequences Kernel is the most general Sequence Kernel. Roughly speaking, it compares two strings by means of the number of contiguous and non-contiguous substrings of a given length they have in common. Non contiguous occurrences are penalized according to the number of gaps they contain.

Formally, let Σ be an alphabet of $|\Sigma|$ symbols, and $s = s_1 s_2 \dots s_{|s|}$ a finite sequence over Σ (i.e. $s_i \in \Sigma, 1 \leqslant i \leqslant |s|$). Let $\mathbf{i} = [i_1, i_2, \dots, i_n]$, with $1 \leqslant i_1 < i_2 < \dots < i_n \leqslant |s|$, be a subset of the indices in s: we will denote as $s[\mathbf{i}] \in \Sigma^n$ the subsequence $s_{i_1} s_{i_2} \dots s_{i_n}$. Note that $s[\mathbf{i}]$ does not necessarily form a contiguous subsequence of s. For example, if s is the sequence "Ronaldo scored the goal" and $\mathbf{i} = [2, 4]$, then $s[\mathbf{i}]$ is "scored goal". The length spanned by $s[\mathbf{i}]$ in s is $l(\mathbf{i}) = i_n - i_1 + 1$. The feature space associated with the Gap-Weighted Subsequences Kernel of length n is indexed by $I = \Sigma^n$, with the embedding given by

$$\phi^n_u(s) = \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}, u \in \Sigma^n, \tag{1}$$

where $\lambda \in]0, 1]$ is the decay factor used to penalize non-contiguous subsequences[1]. The associate kernel is defined as

$$K_n(s,t) = \langle \phi^n(s), \phi^n(t) \rangle = \sum_{u \in \Sigma^n} \phi^n_u(s) \phi^n_u(t). \tag{2}$$

An explicit computation of Equation 2 is unfeasible even for small values of n. To evaluate more efficiently K_n, we use the recursive formulation proposed in (Lodhi et al., 2002; Saunders et al., 2002; Cancedda et al., 2003) based on a dynamic programming implementation. It is reported in the following equations:

$$K'_0(s,t) = 1, \forall s, t, \tag{3}$$

$$K'_i(s,t) = 0, \text{if } \min(|s|,|t|) < i, \tag{4}$$

$$K''_i(s,t) = 0, \text{if } \min(|s|,|t|) < i, \tag{5}$$

$$K''_i(sx,ty) = \begin{cases} \lambda K''_i(sx,t), & \text{if } x \neq y; \\ \lambda K''_i(sx,t) + \lambda^2 K'_{i-1}(s,t), & \text{otherwise.} \end{cases} \tag{6}$$

$$K'_i(sx,t) = \lambda K'_i(s,t) + K''_i(sx,t), \tag{7}$$

$$K_n(s,t) = 0, \text{if } \min(|s|,|t|) < n, \tag{8}$$

$$K_n(sx,t) = K_n(s,t) + \sum_{j:t_j=x} \lambda^2 K'_{n-1}(s, t[1:j-1]), \tag{9}$$

K'_n and K''_n are auxiliary functions with a similar definition as K_n used to facilitate the computation. Based on all definitions above, K_n can be computed in $O(n|s||t|)$. Using the above recursive definition, it turns out that computing all kernel values for subsequences of lengths up to n is not significantly more costly than computing the kernel for n only.

In the rest of the paper we will use the normalised version of the kernel (Equation 10) to keep the values comparable for different values of n and to be independent from the length of the sequences.

$$\hat{K}(s,t) = \frac{K(s,t)}{\sqrt{K(s,s)K(t,t)}}. \tag{10}$$

3 The Syntagmatic Kernel

As stated in Section 1, syntagmatic relations hold among words arranged in a particular temporal order, hence they can be modeled by Sequence Kernels. The Syntagmatic Kernel is defined as a linear combination of Gap-Weighted Subsequences Kernels that operate at word and PoS tag level. In particular, following the approach proposed by Cancedda et al. (2003), it is possible to adapt sequence kernels to operate at word level by instancing the alphabet Σ with the vocabulary $\mathcal{V} = \{w_1, w_2, \dots, w_k\}$. Moreover, we restricted the generic definition of the Gap-Weighted Subsequences Kernel to recognize collocations in the local context of a specified word. The resulting kernel, called *n-gram* Collocation Kernel (K^n_{Coll}), operates on sequences of lemmata around a specified word l_0 (i.e. $l_{-3}, l_{-2}, l_{-1}, l_0, l_{+1}, l_{+2}, l_{+3}$). This formulation allows us to estimate the number of common (sparse) subsequences of lemmata (i.e. collocations) between two examples, in order to capture syntagmatic similarity.

Analogously, we defined the PoS Kernel (K^n_{PoS}) to operate on sequences of PoS tags p_{-3}, p_{-2}, p_{-1}, p_0, p_{+1}, p_{+2}, p_{+3}, where p_0 is the PoS tag of l_0.

The Collocation Kernel and the PoS Kernel are defined by Equations 11 and 12, respectively.

$$K_{Coll}(s,t) = \sum_{l=1}^{n} K^l_{Coll}(s,t) \tag{11}$$

and

$$K_{PoS}(s,t) = \sum_{l=1}^{n} K^l_{PoS}(s,t). \tag{12}$$

Both kernels depend on the parameter n, the length of the non-contiguous subsequences, and λ, the de-

[1] Notice that by choosing $\lambda = 1$ sparse subsequences are not penalized. On the other hand, the kernel does not take into account sparse subsequences with $\lambda \to 0$.

cay factor. For example, K^2_{Coll} allows us to represent all (sparse) bi-grams in the local context of a word.

Finally, the Syntagmatic Kernel is defined as

$$K_{Synt}(s,t) = K_{Coll}(s,t) + K_{PoS}(s,t). \quad (13)$$

We will show that in WSD, the Syntagmatic Kernel is more effective than standard bigrams and trigrams of lemmata and PoS tags typically used as features.

4 Soft-Matching Criteria

In the definition of the Syntagmatic Kernel only exact word matches contribute to the similarity. To overcome this problem, we further extended the definition of the Gap-Weigthed Subsequences Kernel given in Section 2 to allow soft-matching between words. In order to develop soft-matching criteria, we follow the idea that two words can be substituted preserving the meaning of the whole sentence if they are paradigmatically related (e.g. synomyns, hyponyms or domain related words). If the meaning of the proposition as a whole is preserved, the meaning of the lexical constituents of the sentence will necessarily remain unchanged too, providing a viable criterion to define a soft-matching schema. This can be implemented by "plugging" external paradigmatic information into the Collocation kernel.

Following the approach proposed by (Shawe-Taylor and Cristianini, 2004), the soft-matching Gap-Weighted Subsequences Kernel is now calculated recursively using Equations 3 to 5, 7 and 8, replacing Equation 6 by the equation:

$$K''_i(sx,ty) = \lambda K''_i(sx,t) + \lambda^2 a_{xy} K'_{i-1}(s,t), \forall x,y, \quad (14)$$

and modifying Equation 9 to:

$$K_n(sx,t) = K_n(s,t) + \sum_j^{|t|} \lambda^2 a_{xt_j} K'_{n-1}(s, t[1:j-1]). \quad (15)$$

where a_{xy} are entries in a similarity matrix $\mathbf{A}$ between symbols (words). In order to ensure that the resulting kernel is valid, $\mathbf{A}$ must be positive semi-definite.

In the following subsections, we describe two alternative soft-matching criteria based on WordNet Synonymy and Domain Proximity. In both cases, to show that the similarity matrices are a positive semi-definite we use the following result:

Proposition 1 *A matrix* $\mathbf{A}$ *is positive semi-definite if and only if* $\mathbf{A} = \mathbf{B}^T\mathbf{B}$ *for some real matrix* $\mathbf{B}$.

The proof is given in (Shawe-Taylor and Cristianini, 2004).

4.1 WordNet Synonymy

The first solution we have experimented exploits a lexical resource representing paradigmatic relations among terms, i.e. WordNet. In particular, we used WordNet-1.7.1 for English and the Italian part of MultiWordNet[2].

In order to find a similarity matrix between terms, we defined a vector space where terms are represented by the WordNet synsets in which such terms appear. Hence, we can view a term as vector in which each dimension is associated with one synset. The term-by-synset matrix $\mathbf{S}$ is then the matrix whose rows are indexed by the synsets. The entry x_{ij} of $\mathbf{S}$ is 1 if the synset s_j contains the term w_i, and 0 otherwise. The term-by-synset matrix $\mathbf{S}$ gives rise to the similarity matrix $\mathbf{A} = \mathbf{S}\mathbf{S}^T$ between terms. Since $\mathbf{A}$ can be rewritten as $\mathbf{A} = (\mathbf{S}^T)^T\mathbf{S}^T = \mathbf{B}^T\mathbf{B}$, it follows directly by Proposition 1 that it is positive semi-definite.

It is straightforward to extend the soft-matching criterion to include hyponym relation, but we achieved worse results. In the evaluation section we will not report such results.

4.2 Domain Proximity

The approach described above requires a large scale lexical resource. Unfortunately, for many languages, such a resource is not available. Another possibility for implementing soft-matching is introducing the notion of Semantic Domains.

Semantic Domains are groups of strongly paradigmatically related words, and can be acquired automatically from corpora in a totally unsupervised way (Gliozzo, 2005). Our proposal is to exploit a Domain Proximity relation to define a soft-matching criterion on the basis of an unsupervised similarity metric defined in a Domain Space. The Domain Space can be determined once a Domain

[2] http://multiwordnet.itc.it

Model (DM) is available. This solution is evidently cheaper, because large collections of unlabeled texts can be easily found for every language.

A DM is represented by a $k \times k'$ rectangular matrix $\mathbf{D}$, containing the domain relevance for each term with respect to each domain, as illustrated in Table 1. DMs can be acquired from texts by exploiting a lexical coherence assumption (Gliozzo, 2005). To this aim, Term Clustering algorithms can be used: a different domain is defined for each cluster, and the degree of association between terms and clusters, estimated by the unsupervised learning algorithm, provides a domain relevance function. As a clustering technique we exploit Latent Semantic Analysis (LSA), following the methodology described in (Gliozzo et al., 2005b). This operation is done offline, and can be efficiently performed on large corpora.

	MEDICINE	COMPUTER_SCIENCE
HIV	1	0
AIDS	1	0
virus	0.5	0.5
laptop	0	1

Table 1: Example of Domain Model.

LSA is performed by means of SVD of the term-by-document matrix $\mathbf{T}$ representing the corpus. The SVD algorithm can be exploited to acquire a domain matrix $\mathbf{D}$ from a large corpus in a totally unsupervised way. SVD decomposes the term-by-document matrix $\mathbf{T}$ into three matrices $\mathbf{T} = \mathbf{V}\mathbf{\Sigma_k}\mathbf{U}^T$ where $\mathbf{\Sigma_k}$ is the diagonal $k \times k$ matrix containing the k singular values of $\mathbf{T}$. $\mathbf{D} = \mathbf{V}\mathbf{\Sigma_{k'}}$ where $k' \ll k$.

Once a DM has been defined by the matrix $\mathbf{D}$, the Domain Space is a k' dimensional space, in which both texts and terms are represented by means of Domain Vectors (DVs), i.e. vectors representing the domain relevances among the linguistic object and each domain. The DV $\vec{w_i'}$ for the term $w_i \in \mathcal{V}$ is the i^{th} row of $\mathbf{D}$, where $\mathcal{V} = \{w_1, w_2, \ldots, w_k\}$ is the vocabulary of the corpus.

The term-by-domain matrix $\mathbf{D}$ gives rise to the term-by-term similarity matrix $\mathbf{A} = \mathbf{D}\mathbf{D}^T$ among terms. It follows from Proposition 1 that $\mathbf{A}$ is positive semi-definite.

5 Kernel Combination for WSD

To improve the performance of a WSD system, it is possible to combine different kernels. Indeed, we followed this approach in the participation to Senseval-3 competition, reaching the state-of-the-art in many lexical-sample tasks (Strapparava et al., 2004). While this paper is focused on Syntagmatic Kernels, in this section we would like to spend some words on another important component for a complete WSD system: the Domain Kernel, used to model domain relations.

Syntagmatic information alone is not sufficient to define a full kernel for WSD. In fact, in (Magnini et al., 2002), it has been claimed that knowing the domain of the text in which the word is located is a crucial information for WSD. For example the (domain) polysemy among the COMPUTER_SCIENCE and the MEDICINE senses of the word `virus` can be solved by simply considering the domain of the context in which it is located.

This fundamental aspect of lexical polysemy can be modeled by defining a kernel function to estimate the domain similarity among the contexts of the words to be disambiguated, namely the *Domain Kernel*. The Domain Kernel measures the similarity among the topics (domains) of two texts, so to capture domain aspects of sense distinction. It is a variation of the Latent Semantic Kernel (Shawe-Taylor and Cristianini, 2004), in which a DM is exploited to define an explicit mapping $\mathcal{D} : \mathbb{R}^k \rightarrow \mathbb{R}^{k'}$ from the Vector Space Model (Salton and McGill, 1983) into the Domain Space (see Section 4), defined by the following mapping:

$$\mathcal{D}(\vec{t_j}) = \vec{t_j}(\mathbf{I^{IDF}}\mathbf{D}) = \vec{t_j'} \tag{16}$$

where $\mathbf{I^{IDF}}$ is a $k \times k$ diagonal matrix such that $i_{i,i}^{IDF} = IDF(w_i)$, $\vec{t_j}$ is represented as a row vector, and $IDF(w_i)$ is the *Inverse Document Frequency* of w_i. The Domain Kernel is then defined by:

$$K_D(t_i, t_j) = \frac{\langle \mathcal{D}(t_i), \mathcal{D}(t_j) \rangle}{\sqrt{\langle \mathcal{D}(t_j), \mathcal{D}(t_j) \rangle \langle \mathcal{D}(t_i), \mathcal{D}(t_i) \rangle}} \tag{17}$$

The final system for WSD results from a combination of kernels that deal with syntagmatic and paradigmatic aspects (i.e. PoS, collocations, bag of words, domains), according to the following kernel

combination schema:

$$K_C(x_i, x_j) = \sum_{l=1}^{n} \frac{K_l(x_i, x_j)}{\sqrt{K_l(x_j, x_j) K_l(x_i, x_i)}} \quad (18)$$

6 Evaluation

In this section we evaluate the Syntagmatic Kernel, showing that it improves over the standard feature extraction technique based on bigrams and trigrams of words and PoS tags.

6.1 Experimental settings

We conducted the experiments on two lexical sample tasks (English and Italian) of the Senseval-3 competition (Mihalcea and Edmonds, 2004). In lexical-sample WSD, after selecting some target words, training data is provided as a set of texts. For each text a given target word is manually annotated with a sense from a predetermined set of possibilities. Table 2 describes the tasks by reporting the number of words to be disambiguated, the mean polysemy, and the dimension of training, test and unlabeled corpora. Note that the organizers of the English task did not provide any unlabeled material. So for English we used a domain model built from the training partition of the task (obviously skipping the sense annotation), while for Italian we acquired the DM from the unlabeled corpus made available by the organizers.

	#w	*pol*	*# train*	*# test*	*# unlab*
English	57	6.47	7860	3944	7860
Italian	45	6.30	5145	2439	74788

Table 2: Dataset descriptions.

6.2 Performance of the Syntagmatic Kernel

Table 3 shows the performance of the Syntagmatic Kernel on both data sets. As baseline, we report the result of a standard approach consisting on explicit bigrams and trigrams of words and PoS tags around the words to be disambiguated (Yarowsky, 1994). The results show that the Syntagmatic Kernel outperforms the baseline in any configuration (hard/soft-matching). The soft-matching criteria further improve the classification performance. It is interesting to note that the Domain Proximity methodology obtained better results than WordNet Synonymy. The different results observed between Italian and English using the Domain Proximity soft-matching criterion are probably due to the small size of the unlabeled English corpus.

Standard approach		
	English	*Italian*
Bigrams and trigrams	67.3	51.0
Syntagmatic Kernel		
Hard matching	67.7	51.9
Soft matching (WordNet)	67.3	51.3
Soft matching (Domain proximity)	68.5	54.0

Table 3: Performance (F1) of the Syntagmatic Kernel.

In these experiments, the parameters n and λ are optimized by cross-validation. For K^n_{Coll}, we obtained the best results with $n = 2$ and $\lambda = 0.5$. For K^n_{PoS}, $n = 3$ and $\lambda \rightarrow 0$. The domain cardinality k' was set to 50.

Finally, the global performance (F1) of the full WSD system (see Section 5) on English and Italian lexical sample tasks is 73.3 for English and 61.3 for Italian. To our knowledge, these figures represent the current state-of-the-art on these tasks.

7 Conclusion and Future Work

In this paper we presented the Syntagmatic Kernels, i.e. a set of kernel functions that can be used to model syntagmatic relations for a wide variety of Natural Language Processing tasks. In addition, we proposed two soft-matching criteria for the sequence analysis, which can be easily modeled by relaxing the constraints in a Gap-Weighted Subsequences Kernel applied to local contexts of the word to be analyzed. Experiments, performed on two lexical sample Word Sense Disambiguation benchmarks, show that our approach further improves the standard techniques usually adopted to deal with syntagmatic relations. In addition, the Domain Proximity soft-matching criterion allows us to define a semi-supervised learning schema, improving the overall results.

For the future, we plan to exploit the Syntagmatic Kernel for a wide variety of Natural Language Processing tasks, such as Entity Recognition and Relation Extraction. In addition we are applying the soft matching criteria here defined to Tree Kernels,

in order to take into account lexical variability in parse trees. Finally, we are going to further improve the soft-matching criteria here proposed by exploring the use of entailment criteria for substitutability.

Acknowledgments

The authors were partially supported by the Onto-Text Project, funded by the Autonomous Province of Trento under the FUP-2004 research program.

References

N. Cancedda, E. Gaussier, C. Goutte, and J.M. Renders. 2003. Word-sequence kernels. *Journal of Machine Learning Research*, 32(6):1059–1082.

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

A. Gliozzo, C. Giuliano, and R. Rinaldi. 2005a. Instance filtering for entity recognition. *ACM SIGKDD Explorations, special Issue on Natural Language Processing and Text Mining*, 7(1):11–18, June.

A. Gliozzo, C. Giuliano, and C. Strapparava. 2005b. Domain kernels for word sense disambiguation. In *Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL-05)*, pages 403–410, Ann Arbor, Michigan, June.

A. Gliozzo. 2005. *Semantic Domains in Computational Linguistics*. Ph.D. thesis, ITC-irst/University of Trento.

H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(3):419–444.

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. Mihalcea and P. Edmonds, editors. 2004. *Proceedings of SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, July.

G. Salton and M.H. McGill. 1983. *Introduction to modern information retrieval*. McGraw-Hill, New York.

C. Saunders, H. Tschach, and J. Shawe-Taylor. 2002. Syllables and other string kernel extensions. In *Proceedings of 19th International Conference on Machine Learning (ICML02)*.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

C. Strapparava, C. Giuliano, and A. Gliozzo. 2004. Pattern abstraction and term similarity for word sense disambiguation: IRST at Senseval-3. In *Proceedings of SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, July.

D. Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In *Proceedings of the 32nd Annual Meeting of the ACL*, pages 88–95, Las Cruces, New Mexico.