Discriminative Slot Detection Using Kernel Methods

Shubin Zhao, Adam Meyers, Ralph Grishman
Department of Computer Science
New York University
715 Broadway, New York, NY 10003
shubinz, meyers, grishman@cs.nyu.edu

Abstract

Most traditional information extraction approaches are generative models that assume events exist in text in certain patterns and these patterns can be regenerated in various ways. These assumptions limited the syntactic clues being considered for finding an event and confined these approaches to a particular syntactic level. This paper presents a discriminative framework based on kernel SVMs that takes into account different levels of syntactic information and automatically identifies the appropriate clues. Kernels are used to represent certain levels of syntactic structure and can be combined in principled ways as input for an SVM. We will show that by combining a low level sequence kernel with a high level kernel on a GLARF dependency graph, the new approach outperformed a good rule-based system on slot filler detection for MUC-6.

1 Introduction

The goal of Information Extraction (IE) is to extract structured facts of interest from text and present them in databases or templates. Much of the IE research was promoted by the US Government-sponsored MUCs (Message Understanding Conferences). The techniques used by Information Extraction depend greatly on the sublanguage used in a domain, such as financial news or medical records. The training data for an IE system is often sparse since the target domain changes quickly. Traditional IE approaches try to generate patterns for events by various means using training data. For example, the FASTUS (Appelt et al., 1996) and Proteus (Grishman, 1996) systems, which performed well for MUC-6, used hand-written rules for event patterns. The symbolic learning systems, like AutoSlog (Riloff, 1993) and CRYSTAL (Fisher et al., 1996), generated patterns automatically from specific examples (text segments) using generalization and predefined pattern templates. There are also statistical approaches (Miller et al., 1998) (Collins et al., 1998) trying to encode event patterns in statistical CFG grammars. All of these approaches assume events occur in text in certain patterns. However this assumption may not be completely correct and it limits the syntactic information considered by these approaches for finding events, such as information on global features from levels other than deep processing. This paper will show that a simple bag-of-words model can give us reliable information about event occurrence. When training data is limited, these other approaches may also be less effective in their ability to generate reliable patterns.

The idea for overcoming these problems is to avoid making any prior assumption about the syntactic structure an event may assume; instead, we should consider all syntactic features in the target text and use a discriminative classifier to decide that automatically. Discriminative classifiers make no attempt to resolve the structure of the target classes. They only care about the decision boundary to separate the classes. In our case, we only need criteria to predict event elements from text using the syntactic features provided. This seems a more suitable solution for IE where training data is often sparse.

This paper presents an approach that uses kernel functions to represent different levels of syntactic structure (information). With the properties of kernel functions, individual kernels can be combined freely into comprehensive kernels that cross syntactic levels. The classifier we chose to use is SVM (Support Vector Machine), mostly due to its ability to work in high dimensional feature spaces. The experimental results of this approach show that it can outperform a hand-crafted rule system for the MUC-6 management succession domain.

2 Background

2.1 Information Extraction

The major task of IE is to find the elements of an event from text and combine them to form templates or populate databases. Most of these elements are named entities (NEs) involved in the event. To determine which entities in text are involved, we need to find reliable clues around each entity. The extraction procedure starts with

text preprocessing, ranging from tokenization and part-of-speech tagging to NE identification and parsing. Traditional approaches would use various methods of analyzing the results of deep preprocessing to find patterns. Here we propose to use support vector machines to identify clues automatically from the outputs of different levels of preprocessing.

2.2 Support Vector Machine

For a two-class classifier, with separable training data, when given a set of *n* labeled vector examples

$$(X_1, y_1), (X_2, y_2), ..., (X_n, y_n),\ y_i \in \{+1, -1\},$$

a support vector machine (Vapnik, 1998) produces the separating hyperplane with largest margin among all the hyperplanes that successfully classify the examples. Suppose that all the examples satisfy the following constraint:

$$y_i \times (< W, X_i > + b) \geq 1$$

It is easy to see that the margin between the two bounding hyperplanes $< W, X_i > + b = \pm 1$ is $2/||W||$. So maximizing the margin is equivalent to minimizing $||W||^2$ subject to the separation constraint above. In machine learning theory, this margin relates to the upper bound of the VC-dimension of a support vector machine. Increasing the margin reduces the VC-dimension of the learning system, thus increasing the generalization capability of the system. So a support vector machine produces a classifier with optimal generalization capability. This property enables SVMs to work in high dimensional vector spaces.

2.3 Kernel SVM

The vectors in SVM are usually feature vectors extracted by a certain procedure from the original objects, such as images or sentences. Since the only operator used in SVM is the dot product between two vectors, we can replace this operator by a function $\varphi(S_i, S_j)$ on the object domain. In our case, S_i and S_j are sentences. Mathematically this is still valid as long as $\varphi(S_i, S_j)$ satisfies Mercer's condition[1]. Function $\varphi(S_i, S_j)$ is often referred to as a kernel function or just a kernel. Kernel functions provide a way to compute the similarity between two objects without transforming them into features.

The kernel set has the following properties:

1. If $K_1(x, y)$ and $K_2(x, y)$ are kernels on $X \times Y$, $\alpha, \beta > 0$, then $\alpha K_1(x, y) + \beta K_2(x, y)$ is a kernel on $X \times Y$.
2. If $K_1(x, y)$ and $K_2(x, y)$ are kernels on $X \times Y$, then $K_1(x, y) \times K_2(x, y)$ is a kernel on $X \times Y$.
3. If $K_1(x, y)$ is a kernel on $X \times Y$ and $K_2(u, v)$ is a kernel on $U \times V$, then $K((x, u), (y, v)) = K_1(x, y) + K_2(u, v)$ is a kernel on $(X \times U) \times (Y \times V)$.

When we have kernels representing information from different sources, these properties enable us to incorporate them into one kernel. The general kernels such as RBF or polynomial kernels (Müller et al., 2001), which extend features nonlinearly into higher dimensional space, can also be applied to either the combination kernel or to each component kernel individually.

2.4 Related Work

There have been a number of SVM applications in NLP using particular levels of syntactic information. (Lodhi et al., 2002) compared a word-based string kernel and n-gram kernels at the sequence level for a text categorization task. The experimental results showed that the n-gram kernels performed quite well for the task. Although string kernels can capture common word subsequences with gaps, its geometric penalty factor may not be suitable for weighting the long distance features. (Collins et al., 2001) suggested kernels on parse trees and other structures for general NLP tasks. These kernels count small subcomponents multiple times so that in practice one has to be careful to avoid overfitting. This can be achieved by limiting the matching depth or using a penalty factor to downweight large components.

(Zelenko et al., 2003) devised a kernel on shallow parse trees to detect relations between named entities, such as the person-affiliation relation between a person name and an organization name. The so-called relation kernel matches from the roots of two trees and continues recursively to the leaf nodes if the types of two nodes match.

All the kernels used in these works were applied to a particular syntactic level. This paper presents an approach for information extraction that uses kernels to combine information from different levels and automatically identify which information contributes to the task. This framework can also be applied to other NLP tasks.

[1] The matrix must be positive semi-definite

3 A Discriminative Framework

The discriminative framework proposed here is called ARES (Automated Recognition of Event Slots). It makes no assumption about the text structure of events. Instead, kernels are used to represent syntactic information from various syntactic sources. The structure of ARES is shown in Fig 1. The preprocessing modules include a part-of-speech tagger, name tagger, sentence parser and GLARF parser, but are not limited to these. Other general tools can also be included, which are not shown in the diagram. The triangles in the diagram are kernels that encode the corresponding syntactic processing result. In the training phase, the target slot fillers are labeled in the text so that SVM slot detectors can be trained through the kernels to find fillers for the key slots of events. In the testing phase, the SVM classifier will predict the slot fillers from unlabeled text and a merging procedure will merge slots into events if necessary. The main kernel we propose to use is on GLARF (Meyers et al., 2001) dependency graphs.

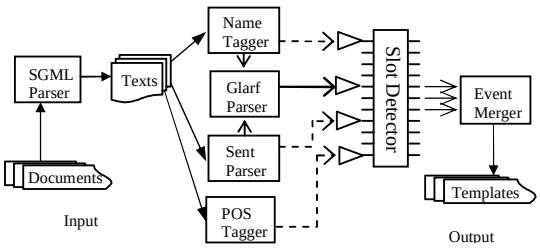

Fig 1. Structure of the discriminative model

The idea is that an IE model should not commit itself to any syntactic level. The low level information, such as word collocations, may also give us important clues. Our experimentation will show that for the MUC-6 management succession domain, even bag-of-words or n-grams can give us helpful information about event occurrence.

3.1 Syntactic Kernels

To make use of syntactic information from different levels, we can develop kernel functions or syntactic kernels to represent a certain level of syntactic structure. The possible syntactic kernels include

- Sequence kernels: representing sequence level information, such as bag-of-words, n-grams, string kernel, etc.
- Phrase kernel: representing information at an intermediate level, such as kernels based on multiword expressions, chunks or shallow parse trees.
- Parsing kernel: representing detailed syntactic structure of a sentence, such as kernels based on parse trees or dependency graphs.

These kernels can be used alone or combined with each other using the properties of kernels. They can also be combined with high-order kernels like polynomial or RBF kernels, either individually or on the resulting kernel.

As the depth of analysis of the preprocessing increases, the accuracy of the result decreases. Combining the results of deeper processing with those of shallower processing (such as n-grams) can also give us a back-off ability to recover from errors in deep processing.

In practice each kernel can be tested for the task as the sole input to an SVM to determine if this level of information is helpful or not. After figuring out all the useful kernels, we can try to combine them to make a comprehensive kernel as final input to the classifier. The way to combine them and the parameters in combination can be determined using validation data.

4 Introduction to GLARF

GLARF (Grammatical and Logical Argument Regularization Framework) [Meyers et al., 2001] is a hand-coded system that produces comprehensive word dependency graphs from Penn TreeBank-II (PTB-II) parse trees to facilitate applications like information extraction. GLARF is designed to enhance PTB-II parsing to produce more detailed information not provided by parsing, such as information about object, indirect object and appositive relations. GLARF can capture more regularization in text by transforming non-canonical (passive, filler-gap) constructions into their canonical forms (simple declarative clauses). This is very helpful for information extraction where training data is often sparse. It also represents all syntactic phenomena in uniform typed PRED-ARG structures, which is convenient for computational purposes. For a sentence, GLARF outputs depencency triples derived automatically from the GLARF typed feature structures [Meyers et al., 2001]. A directed dependency graph of the sentence can also be constructed from the depencency triples. The following is the output of GLARF for the sentence "*Tom Donilon, who also could get a senior job …*".

<SBJ, get, Tom Donilon>
<OBJ, get, job>
<ADV, get, also>
<AUX, get, could>
<T-POS, job, a>

<A-POS, job, senior>

. . .

GLARF can produce logical relations in addition to surface relations, which is helpful for IE tasks. It can also generate output containing the base form of words so that different tenses of verbs can be regularized. Because of all these features, our main kernels are based on the GLARF dependency triples or dependency graphs.

5 Event and Slot Kernels

Here we will introduce the kernels used by ARES for event occurrence detection (EOD) and slot filler detection (SFD).

5.1 EOD Kernels

In Information Extraction, one interesting issue is event occurrence detection, which is determining whether a sentence contains an event occurrence or not. If this information is given, it would be much easier to find the relevant entities for an event from the current sentence or surrounding sentences. Traditional approaches do matching (for slot filling) on all sentences, even though most of them do not contain any event at all. Event occurrence detection is similar to sentence level information retrieval, so simple models like bag-of-words or n-grams could work well. We tried two kernels to do this, one is a sequence level n-gram kernel and the other is a GLARF-based kernel that matches syntactic details between sentences. In the following formulae, we will use an identity function $I(x, y)$ that gives 1 when $x \equiv y$ and 0 otherwise, where x and y are strings or vectors of strings.

1. N-gram kernel $\varphi_N(S_1, S_2)$ that counts common n-grams between two sentences. Given two sentence: $S_1 = <w_1, w_2, .. w_{N_1}>$, and $S_1 = <w_1, w_2, .. w_{N_2}>$, a bigram kernel $\varphi_{bi}(S_1, S_2)$ is

$$\sum_{i=1}^{N_1-1} \sum_{j=1}^{N_2-1} I(< w_i, w_{i+1} >, < w_j, w_{j+1} >).$$

Kernels can be inclusive, in other words, the trigram kernel includes bigrams and unigrams. For the unigram kernel a stop list is used that removes words other than nouns, verbs, adjectives and adverbs.

2. Glarf kernel $\varphi_g(G_1, G_2)$: this kernel is based on the GLARF dependency result. Given the triple outputs of two sentences produced by GLARF: $G_1 = \{< r_i, p_i, a_i >\}$, $1 \le i \le N_1$ and $G_2 = \{< r_j, p_j, a_j >\}$, $1 \le j \le N_2$, where r_i, p_i, a_i correspond to the role label, predicate word and argument word respectively in GLARF output, it matches the two triples, their predicates and arguments respectively. So $\varphi_g(G_1, G_2)$ equals

$$\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} (I(< r_i, p_i, a_i >, < r_j, p_j, a_j >) + \alpha I(p_i, p_j) + \beta I(a_i, a_j))$$

In our experiments, α and β were set to 1.

5.2 SFD Kernels

Slot filler detection (SFD) is the task of determining which named entities fill a slot in some event template. Two kernels were proposed for SFD: the first one matches local contexts of two target NEs, while the second one combines the first one with an n-gram EOD kernel.

1. $\varphi^1_{SFD}(G_i, G_j)$: This kernel was also defined on a GLARF dependency graph (DG), a directed graph constructed from its typed PRED-ARG outputs. The arcs labeled with roles go from predicate words to argument words. This kernel matches local context surrounding a name in a GLARF dependency graph. In preprocessing, all the names of the same type are translated into one symbol (a special word). The matching starts from two anchor nodes (NE nodes of the same type) in the two DG's and recursively goes from these nodes to their successors and predecessors, until the words associated with nodes do not match. In our experiment, the matching depth was set to 2. Each node *n* contains a predicate word *w* and relation pairs $\{< r_i, a_i >\}$, $1 \le i \le p$ representing its *p* arguments and the roles associated with them.

A matching function $C(n_1, n_2)$ is defined as

$$\sum_{i=1}^{p_1} \sum_{j=1}^{p_2} (I(< r_i, a_i >, < r_j, a_j >) + I(r_i, r_j)).$$

Then $\varphi^1_{SFD}(G_i, G_j)$: can be written as

$$C(E_i, E_j) + \sum_{\substack{n_i \in Succ(E_i) \\ n_j \in Succ(E_j) \\ n_i \equiv n_j}} C(n_i, n_j) + \sum_{\substack{n_i \in Pred(E_i) \\ n_j \in Pred(E_j) \\ n_i \equiv n_j}} C(n_i, n_j)$$

where E_i and E_j are the anchor nodes in the two DG's; $n_i \equiv n_j$ is true if the predicate words associated with them match. Functions Succ(*n*) and Pred(*n*) give the successor and predecessor node set of a node *n*. The reason for setting a depth limit is that it covers most of the local syntax of a node (before matching stops); another reason is that the cycles currently present in GLARF dependency graph prohibit unbounded recursive matching.

2. $\varphi^2_{SFD}(S_i, S_j)$: This kernel combines linearly the n-gram event kernel and the slot kernel above,

in the hope that the general event occurrence information provided by EOD kernel can help the slot kernel to ignore NEs in sentences that do not contain any event occurrence.

$\varphi^2{}_{SFD}(S_i, S_j) = \alpha\varphi_N(S_i, S_j) + \beta\varphi^1{}_{SFD}(G_i, G_j)$,

where α, β were set to be 1 in our experiments. The Glarf event kernel was not used, simply because it uses information from the same source as $\varphi^1{}_{SFD}(G_i, G_j)$. The n-gram kernel was chosen to be the trigram kernel, which gives us the best EOD performance among n-gram kernels.

We also tried the dependency graph kernel proposed by (Collins et al., 2001), but it did not give us better result.

6 Experiments

6.1 Corpus

The experiments of ARES were done on the MUC-6 corporate management succession domain using the official training data and, for the final experiment, the official test data as well. The training data was split into a training set (80%) and validation set (20%). In ARES, the text was preprocessed by the Proteus NE tagger and Charniak sentence parser. Then the GLARF processor produced dependency graphs based on the parse trees and NE results. All the names were transformed into symbols representing their types, such as #PERSON# for all person names. The reason is that we think the name itself does not provide a significant clue; the only thing that matters is what type of name occurs at certain position.

Two tasks have been tried: one is EOD (event occurrence detection) on sentences; the other is SFD (slot filler detection) on named entities, including person names and job titles. EOD is to determine whether a sentence contains an event or not. This would give us general information about sentence-level event occurrences. SFD is to find name fillers for event slots. The slots we experimented with were the person name and job title slots in MUC-6. We used the SVM package SVM^{light} in our experiments, embedding our own kernels as custom kernels.

6.2 EOD Experiments

In this experiment, ARES was trained on the official MUC-6 training data to do event occurrence detection. The data contains 1940 sentences, of which 158 are labeled as positive instances (contain an event). Five-fold cross validation was used so that the training and test set contain 80% and 20% of the data respectively. Three kernels defined in the previous section were tried. Table 1 shows the performance of each kernel. Three n-gram kernels were tested: unigram, bigram and trigram. Subsequences longer than trigrams were also tried, but did not yield better results.

The results show that the trigram kernel performed the best among n-gram kernels. GLARF kernel did better than n-gram kernels, which is reasonable because it incorporates detailed syntax of a sentence. But generally speaking, the n-gram kernels alone performed fairly well for this task, which indicates that low level text processing can also provide useful information. The mix kernel that combines the trigram kernel with GLARF kernel gave the best performance, which might indicate that the low level information provides additional clues or helps to overcome errors in deep processing.

Kernel	Precision	Recall	F-score
Unigram	66.0%	66.5%	66.3%
Bigram	73.9%	60.3%	66.4%
Trigram	77.5%	61.5%	68.6%
GLARF	77.5%	63.9%	70.1%
Mix	81.5%	66.4%	73.2%

Table 1. EOD performance of ARES using different kernels. The Mix kernel is a linear combination of the trigram kernel and the Glarf kernel.

6.3 SFD Experiments

The slot filler detection (SFD) task is to find the named entities in text that can fill the corresponding slots of an event.[2] We treat job title as a named entity throughout this paper, although it is not included in the traditional MUC named entity set. The slots we used for evaluation were PERSON_IN (the person who took a position), PERSON_OUT (the person who left a position) and POST (the position involved). We generated the two person slots from the official MUC-6 templates and the corresponding filler strings in text were labeled. Three SVM predictors were trained to find name fillers of each slot. Two experiments have been tried on MUC-6 training data using five-fold cross validation.

The first experiment of ARES used slot kernel $\varphi^1{}_{SFD}(G_i, G_j)$ alone, relying solely on local

[2] We used this task for evaluation, rather than the official MUC template-filling task, in order to assess the system's ability to identify slot fillers separately from its ability to combine them into templates.

context around a NE. From the performance table (Table 2), we can see that local context can give a fairly good clue for finding PERSON_IN and POST, but not for PERSON_OUT. The main reason is that local context might be not enough to determine a PERSON_OUT filler. It often requires inference or other semantic information. For example, the sentence *"Aaron Spelling, the company's vice president, was named president."*, indicates that "Aaron Spelling" left the position of vice president, therefore it should be a PERSON_OUT. But the sentence *"Aaron Spelling, the company's vice president, said …"*, which is very similar to first one in syntax, has no such indication at all. In complicated cases, a person can even hold two positions at the same time.

Accuracy	Precision	Recall	F-score
PER_IN	63.6%	62.5%	63.1%
PER_OUT	54.8%	54.2%	54.5%
POST	64.4%	55.2%	59.4%

Table 2. SFD performance of ARES using kernel $\varphi^1{}_{SFD}(G_i, G_j)$.

In this experiment, the SVM predictor considered all the names identified by the NE tagger; however, most of the sentences do not contain an event occurrence at all, so NEs in these sentences should be ignored no matter what their local context is. To achieve this we need general information about event occurrence, and this is just what the EOD kernel can provide. In our second experiment, we tested the kernel $\varphi^2{}_{SFD}(S_i, S_j)$, which is a linear combination of the trigram EOD kernel and the SFD kernel $\varphi^1{}_{SFD}(G_i, G_j)$. Table 3 shows the performance of the combination kernel, from which we can see that there is clear performance improvement for all three slots. We also tried to use the mix kernel which gave us the best EOD performance, but it did not yield a better result. The reason we think is that the GLARF EOD kernel and SFD kernel are from the same syntactic source, so the information was repeated.

Accuracy	Precision	Recall	F-score
PER_IN	86.6%	60.5%	71.2%
PER_OUT	69.2%	58.2%	63.2%
POST	68.5%	68.9%	68.7%

Table 3. SFD performance of ARES using kernel $\varphi^2{}_{SFD}(S_i, S_j)$. It combines the Glarf SFD kernel with trigram EOD kernel. For PER_OUT, unigram EOD kernel was used.

Since five-fold cross validation was used, ARES was trained on 80% of the MUC-6 training data in these two experiments.

6.4 Comparison with MUC-6 System

This experiment was done on the official MUC-6 training and test data, which contain 50K words and 40K words respectively. ARES used the official corpora as training and test sets, except that in the training data, all the slot fillers were manually labeled. We compared the performance of ARES with the NYU Proteus system, a rule-based system that performed well for MUC-6. To score the performance for these three slots, we generated the slot-filler pairs as keys for a document from the official MUC-6 templates and removed duplicate pairs. The scorer matches the filler string in the response file of ARES to the keys. The response result for Proteus was extracted in the same way from its template output. Table 4. shows the result of ARES using the combination kernel in the previous experiment.

Accuracy	Precision	Recall	F-score
PER_IN	77.3%	62.2%	68.9%
PER_OUT	58.9%	69.7%	63.9%
POST	77.1%	71.5%	73.6%

Table 4. Slot performance ARES using kernel $\varphi^2{}_{SFD}(S_i, S_j)$ on MUC-6 test data.

Table 5 shows the test result of the Proteus system. Comparing the numbers we can see that for slot PERSON_IN and POST, ARES outperformed the Proteus system by a few points. The result is promising considering that this model is fully automatic and does not involve any post-processing. As for the PERSON_OUT slot, the performance of ARES was not as good. As we have discussed before, relying purely on syntax might not help us much; we may need an inference model to resolve this problem.

Accuracy	Precision	Recall	F-score
PER_IN	85.7%	51.2%	64.1%
PER_OUT	78.4%	58.6%	67.1%
POST	83.3%	59.7%	69.5%

Table 5. Slot performance of the rule-based Proteus system for MUC-6.

7 Related Work

(Chieu et al., 2003) reported a feature-based SVM system (ALICE) to extract MUC-4 events of

terrorist attacks. The Alice-ME system demonstrated competitive performance with rule-based systems. The features used by Alice are mainly from parsing. Comparing with ALICE, our system uses kernels on dependency graphs to replace explicit features, an approach which is fully automatic and requires no enumeration of features. The model we proposed can combine information from different syntactic levels in principled ways. In our experiments, we used both word sequence information and parsing level syntax information. The training data for ALICE contains 1700 documents, while for our system it is just 100 documents. When data is sparse, it is more difficult for an automatic system to outperform a rule-based system that incorporates general knowledges.

8 Discussion and Further Works

This paper describes a discriminative approach that can use syntactic clues automatically for slot filler detection. It outperformed a hand-crafted system on sparse data by considering different levels of syntactic clues. The result also shows that low level syntactic information can also come into play in finding events, thus it should not be ignored in the IE framework.

For slot filler detection, several classifiers were trained to find names for each slot and there is no correlation among these classifiers. However, entity slots in events are often strongly correlated, for example the PER_IN and POST slots for management succession events. Since these classifiers take the same input and produce different results, correlation models can be used to integrate these classifiers so that the identification of slot fillers might benefit each other.

It would also be interesting to experiment with the tasks that are more difficult for pattern matching, such as determining the on-the-job status property in MUC-6. Since events often span multiple sentences, another direction is to explore cross-sentence models, which is difficult for traditional approaches. For our approach it is possible to extend the kernel from one sentence to multiple sentences, taking into account the correlation between NE's in adjacent sentences.

9 Acknowledgements

This research was supported in part by the Defense Advanced Research Projects Agency as part of the TIDES program, under Grant N66001-001-1-8917 from the Space and Naval Warfare Systems Center, San Diego, and by the National Science Foundation under Grant ITS-0325657. This paper does not necessarily reflect the position of the U.S. Government.

References

D. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Meyers, and M. Tyson 1996. *SRI International FASTUS system: MUC-6 test results and analysis.* In Proceedings of the Sixth Message Understanding Conference.

H. L. Chieu, H. T. Ng, & Y. K. Lee. 2003. *Closing the Gap: Learning-Based Information Extraction Rivaling Knowledge-Engineering Methods.* In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics.

M. Collins and S. Miller. 1998. *Semantic Tagging using a Probabilistic Context Free Grammar,* In Proceedings of the Sixth Workshop on Very Large Corpora.

M. Collins and N. Duffy. 2001. *Convolution Kernels for Natural Language,* Advances in Neural Information Processing Systems 14, MIT Press.

D. Fisher, S. Soderland, J. McCarthy, F. Feng and W. Lehnert. 1996. *Description of The UMass System As Used For MUC-6.* In Proceedings of the Sixth Message Understanding Conference.

R. Grishman. 1996. *The NYU System for MUC-6 or Where's the Syntax?.* In Proceedings of the Sixth Message Understanding Conference.

H. Lodhi, C. Sander, J. Shawe-Taylor, N. Christianini and C. Watkins. 2002. *Text Classification using String Kernels*. Journal of Machine Learning Research.

A. Meyers, R. Grishman, M. Kosaka and S. Zhao. 2001. *Covering Treebanks with GLARF*. In Proceedings of of the ACL Workshop on Sharing Tools and Resources.

S. Miller, M. Crystal, H. Fox, L. Ramshaw, R. Schwartz, R. Stone, and R. Weischedel. 1998. *BBN: Description of The SIFT System As Used For MUC-7,* In Proceedings of the Seventh Message Understanding Conference.

K.-R. Müller, S. Mika, G. Ratsch, K. Tsuda, B. Scholkopf. 2001. *An introduction to kernel-based learning algorithms,* IEEE Trans. Neural Networks, 12, 2, pages 181-201.

E. Riloff. 1993. *Automatically constructing a dictionary for information extraction tasks*. In Proceedings of the 11th National Conference on Artificial Intelligence, 811-816.

V. N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience Publication.

D. Zelenko, C. Aone and A. Richardella. 2003. *Kernel methods for relation extraction.* Journal of Machine Learning Research.