Semi-Supervised Classification for Extracting Protein Interaction Sentences using Dependency Parsing

Güneş Erkan
University of Michigan
gerkan@umich.edu

Arzucan Özgür
University of Michigan
ozgur@umich.edu

Dragomir R. Radev
University of Michigan
radev@umich.edu

Abstract

We introduce a relation extraction method to identify the sentences in biomedical text that indicate an interaction among the protein names mentioned. Our approach is based on the analysis of the paths between two protein names in the dependency parse trees of the sentences. Given two dependency trees, we define two separate similarity functions (kernels) based on cosine similarity and edit distance among the paths between the protein names. Using these similarity functions, we investigate the performances of two classes of learning algorithms, Support Vector Machines and k-nearest-neighbor, and the semi-supervised counterparts of these algorithms, transductive SVMs and harmonic functions, respectively. Significant improvement over the previous results in the literature is reported as well as a new benchmark dataset is introduced. Semi-supervised algorithms perform better than their supervised version by a wide margin especially when the amount of labeled data is limited.

1 Introduction

Protein-protein interactions play an important role in vital biological processes such as metabolic and signaling pathways, cell cycle control, and DNA replication and transcription (Phizicky and Fields, 1995). A number of (mostly manually curated) databases such as MINT (Zanzoni et al., 2002), BIND (Bader et al., 2003), and SwissProt (Bairoch and Apweiler, 2000) have been created to store protein interaction information in structured and standard formats. However, the amount of biomedical literature regarding protein interactions is increasing rapidly and it is difficult for interaction database curators to detect and curate protein interaction information manually. Thus, most of the protein interaction information remains hidden in the text of the papers in the biomedical literature. Therefore, the development of information extraction and text mining techniques for automatic extraction of protein interaction information from free texts has become an important research area.

In this paper, we introduce an information extraction approach to identify sentences in text that indicate an interaction relation between two proteins. Our method is different than most of the previous studies (see Section 2) on this problem in two aspects: First, we generate the dependency parses of the sentences that we analyze, making use of the dependency relationships among the words. This enables us to make more syntax-aware inferences about the roles of the proteins in a sentence compared to the classical pattern-matching information extraction methods. Second, we investigate semi-supervised machine learning methods on top of the dependency features we generate. Although there have been a number of learning-based studies in this domain, our methods are the first semi-supervised efforts to our knowledge. The high cost of labeling free text for this problem makes semi-supervised methods particularly valuable.

We focus on two semi-supervised learning methods: transductive SVMs (TSVM) (Joachims, 1999),

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 228–237, Prague, June 2007.

and harmonic functions (Zhu et al., 2003). We also compare these two methods with their supervised counterparts, namely SVMs and k-nearest neighbor algorithm. Because of the nature of these algorithms, we propose two similarity functions (*kernels* in SVM terminology) among the instances of the learning problem. The instances in this problem are natural language sentences with protein names in them, and the similarity functions are defined on the positions of the protein names in the corresponding parse trees. Our motivating assumption is that the *path* between two protein names in a dependency tree is a good description of the semantic relation between them in the corresponding sentence. We consider two similarity functions; one based on the cosine similarity and the other based on the edit distance among such paths.

2 Related Work

There have been many approaches to extract protein interactions from free text. One of them is based on matching pre-specified patterns and rules (Blaschke et al., 1999; Ono et al., 2001). However, complex cases that are not covered by the pre-defined patterns and rules cannot be extracted by these methods. Huang *et al.* (2004) proposed a method where patterns are discovered automatically from a set of sentences by dynamic programming. Bunescu *et al.* (2005) have studied the performance of rule learning algorithms. They propose two methods for protein interaction extraction. One is based on the rule learning method Rapier and the other on longest common subsequences. They show that these methods outperform hand-written rules.

Another class of approaches is using more syntax-aware natural language processing (NLP) techniques. Both full and partial (shallow) parsing strategies have been applied in the literature. In partial parsing the sentence structure is decomposed partially and local dependencies between certain phrasal components are extracted. An example of the application of this method is relational parsing for the *inhibition* relation (Pustejovsky et al., 2002). In full parsing, however, the full sentence structure is taken into account. Temkin and Gilder (2003) used a full parser with a lexical analyzer and a context free grammar (CFG) to extract protein-protein interaction from text. Another study that uses full-sentence parsing to extract human protein interactions is (Daraselia et al., 2004). Alternatively, Yakushiji *et al.* (2005) propose a system based on head-driven phrase structure grammar (HPSG). In their system protein interaction expressions are presented as predicate argument structure patterns from the HPSG parser. These parsing approaches consider only syntactic properties of the sentences and do not take into account semantic properties. Thus, although they are complicated and require many resources, their performance is not satisfactory.

Machine learning techniques for extracting protein interaction information have gained interest in the recent years. The PreBIND system uses SVM to identify the existence of protein interactions in abstracts and uses this type of information to enhance manual expert reviewing for the BIND database (Donaldson et al., 2003). Words and word bigrams are used as binary features. This system is also tested with the Naive Bayes classifier, but SVM is reported to perform better. Mitsumori *et al.* (2006) also use SVM to extract protein-protein interactions. They use bag-of-words features, specifically the words around the protein names. These systems do not use any syntactic or semantic information. Sugiyama *et al.* (2003) extract features from the sentences based on the verbs and nouns in the sentences such as the verbal forms, and the part of speech tags of the 20 words surrounding the verb (10 before and 10 after it). Further features are used to indicate whether a noun is found, as well as the part of speech tags for the 20 words surrounding the noun, and whether the noun contains numerical characters, non-alpha characters, or uppercase letters. They construct k-nearest neighbor, decision tree, neural network, and SVM classifiers by using these features. They report that the SVM classifier performs the best. They use part-of-speech information, but do not consider any dependency or semantic information.

The paper is organized as follows. In Section 3 we describe our method of extracting features from the dependency parse trees of the sentences and defining the similarity between two sentences. In Section 4 we discuss our supervised and semi-supervised methods. In Section 5 we describe the data sets and evaluation metrics that we used, and present our results. We conclude in Section 6.

3 Sentence Similarity Based on Dependency Parsing

In order to apply the semi-supervised harmonic functions and its supervised counterpart kNN, and the kernel based TSVM and SVM methods, we need to define a similarity measure between two sentences. For this purpose, we use the dependency parse trees of the sentences. Unlike a syntactic parse (which describes the syntactic constituent structure of a sentence), the dependency parse of a sentence captures the semantic predicate-argument relationships among its words. The idea of using dependency parse trees for relation extraction in general was studied by Bunescu and Mooney (2005a). To extract the relationship between two entities, they design a kernel function that uses the shortest path in the dependency tree between them. The motivation is based on the observation that the shortest path between the entities usually captures the necessary information to identify their relationship. They show that their approach outperforms the dependency tree kernel of Culotta and Sorensen (2004), which is based on the subtree that contains the two entities. We adapt the idea of Bunescu and Mooney (2005a) to the task of identifying protein-protein interaction sentences. We define the similarity between two sentences based on the paths between two proteins in the dependency parse trees of the sentences.

In this study we assume that the protein names have already been annotated and focus instead on the task of extracting protein-protein interaction sentences for a given protein pair. We parse the sentences with the Stanford Parser[1] (de Marneffe et al., 2006). From the dependency parse trees of each sentence we extract the shortest path between a protein pair.

For example, Figure 1 shows the dependency tree we got for the sentence "*The results demonstrated that KaiC interacts rhythmically with KaiA, KaiB, and SasA*." This example sentence illustrates that the dependency path between a protein pair captures the relevant information regarding the relationship between the proteins better compared to using the words in the unparsed sentence. Consider the protein pair KaiC and SasA. The words in the sentence between these proteins are *interacts, rhythmically, with, KaiA, KaiB,* and *and.* Among these words *rhythmically, KaiA, and* and *KaiB* are not directly related to the interaction relationship between KaiC and SasA. On the other hand, the words in the dependency path between this protein pair give sufficient information to identify their relationship.

In this sentence we have four proteins (KaiC, KaiA, KaiB, and SasA). So there are six pairs of proteins for which a sentence may or may not be describing an interaction. The following are the paths between the six protein pairs. In this example there is a single path between each protein pair. However, there may be more than one paths between a protein pair, if one or both appear multiple times in the sentence. In such cases, we select the shortest paths between the protein pairs.

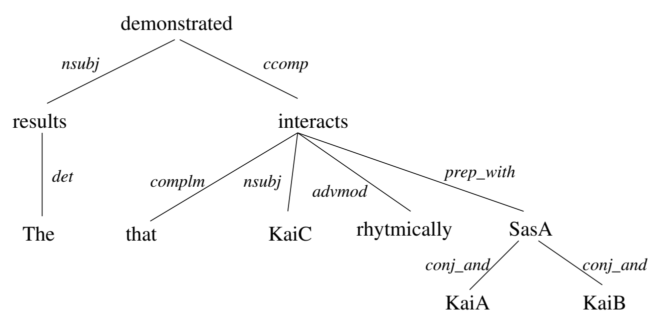

Figure 1: The dependency tree of the sentence "*The results demonstrated that KaiC interacts rhythmically with KaiA, KaiB, and SasA*."

1. KaiC - nsubj - interacts - prep_with - SasA
2. KaiC - nsubj - interacts - prep_with - SasA - conj_and - KaiA
3. KaiC - nsubj - interacts - prep_with – SasA - conj_and - KaiB
4. SasA - conj_and - KaiA
5. SasA - conj_and - KaiB
6. KaiA – conj_and – SasA - conj_and - KaiB

If a sentence contains n different proteins, there are $\binom{n}{2}$ different pairs of proteins. We use machine learning approaches to classify each sentence as an interaction sentence or not for a protein pair. A sentence may be an interaction sentence for one protein

[1] http://nlp.stanford.edu/software/lex-parser.shtml

pair, while not for another protein pair. For instance, our example sentence is a positive interaction sentence for the *KaiC* and *SasA* protein pair. However, it is a negative interaction sentence for the *KaiA* and *SasA* protein pair, i.e., it does not describe an interaction between this pair of proteins. Thus, before parsing a sentence, we make multiple copies of it, one for each protein pair. To reduce data sparseness, we rename the proteins in the pair as *PROTX1* and *PROTX2*, and all the other proteins in the sentence as *PROTX0*. So, for our example sentence we have the following instances in the training set:

1. *PROTX1* - nsubj - interacts - prep_with - *PROTX2*
2. *PROTX1* - nsubj - interacts - prep_with - *PROTX0* - conj_and - *PROTX2*
3. *PROTX1* - nsubj - interacts - prep_with – *PROTX0* - conj_and - *PROTX2*
4. *PROTX1* - conj_and - *PROTX2*
5. *PROTX1* - conj_and - *PROTX2*
6. *PROTX1* – conj_and – *PROTX0* - conj_and - *PROTX2*

The first three instances are positive as they describe an interaction between *PROTX1* and *PROTX2*. The last three are negative, as they do not describe an interaction between *PROTX1* and *PROTX2*.

We define the similarity between two instances based on cosine similarity and edit distance based similarity between the paths in the instances.

3.1 Cosine Similarity

Suppose p_i and p_j are the paths between *PROTX1* and *PROTX2* in instance x_i and instance x_j, respectively. We represent p_i and p_j as vectors of term frequencies in the vector-space model. The cosine similarity measure is the cosine of the angle between these two vectors and is calculated as follows:

$$cos\_sim(p_i, p_j) = cos(\mathbf{p_i}, \mathbf{p_j}) = \frac{\mathbf{p_i} \bullet \mathbf{p_j}}{\|\mathbf{p_i}\|\|\mathbf{p_j}\|} \quad (1)$$

that is, it is the dot product of $\mathbf{p_i}$ and $\mathbf{p_j}$ divided by the lengths of $\mathbf{p_i}$ and $\mathbf{p_j}$. The cosine similarity measure takes values in the range $[0, 1]$. If all the terms in p_i and p_j are common, then it takes the maximum value of 1. If none of the terms are common, then it takes the minimum value of 0.

3.2 Similarity Based on Edit Distance

A shortcoming of cosine similarity is that it only takes into account the common terms, but does not consider their order in the path. For this reason, we also use a similarity measure based on edit distance (also called Levenshtein distance). Edit distance between two strings is the minimum number of operations that have to be performed to transform the first string to the second. In the original character-based edit distance there are three types of operations. These are insertion, deletion, or substitution of a single character. We modify the character-based edit distance into a word-based one, where the operations are defined as insertion, deletion, or substitution of a single word.

The edit distance between path 1 and path 2 of our example sentence is 2. We insert *PROTX0* and *conj_and* to path 1 to convert it to path 2.

1. *PROTX1* - nsubj - interacts - prep_with - **insert (*PROTX0*)** - **insert (*conj_and*)** – *PROTX2*
2. *PROTX1* - nsubj - interacts - prep_with - *PROTX0* - conj_and - *PROTX2*

We normalize edit distance by dividing it by the length (number of words) of the longer path, so that it takes values in the range $[0, 1]$. We convert the distance measure into a similarity measure as follows.

$$edit\_sim(p_i, p_j) = e^{-\gamma(edit\_distance(p_i, p_j))} \quad (2)$$

Bunescu and Mooney (2005a) propose a similar method for relation extraction in general. However, their similarity measure is based on the number of the overlapping words between two paths. When two paths have different lengths, they assume the similarity between them is zero. On the other hand, our edit distance based measure can also account for deletions and insertions of words.

4 Semi-Supervised Machine Learning Approaches

4.1 kNN and Harmonic Functions

When a similarity measure is defined among the instances of a learning problem, a simple and natural choice is to use a nearest neighbor based approach that classifies each instance by looking at the labels of the instances that are most similar to it. Perhaps the simplest and most popular similarity-based learning algorithm is the k-nearest neighbor classification method (kNN). Let U be the set of unlabeled instances, and L be the set of labeled instances in a learning problem. Given an instance $x \in U$, let $N_k^L(x)$ be the set of top k instances in L that are most similar to x with respect to some similarity measure. The kNN equation for a binary classification problem can be written as:

$$y(x) = \sum_{z \in N_k^L(x)} \frac{sim(x,z)y(z)}{\sum_{z' \in N_k^L(x)} sim(x,z')} \quad (3)$$

where $y(z) \in \{0,1\}$ is the label of the instance z.[2] Note that $y(x)$ can take any real value in the $[0,1]$ interval. The final classification decision is made by setting a threshold in this interval (e.g. 0.5) and classifying the instances above the threshold as positive and others as negative. For our problem, each instance is a dependency path between the proteins in the pair and the similarity function can be one of the functions we have defined in Section 3.

Equation 3 can be seen as averaging the labels (0 or 1) of the nearest neighbors of each unlabeled instance. This suggests a generalized semi-supervised version of the same algorithm by incorporating unlabeled instances as neighbors as well:

$$y(x) = \sum_{z \in N_k^{L \cup U}(x)} \frac{sim(x,z)y(z)}{\sum_{z' \in N_k^{L \cup U}(x)} sim(x,z')} \quad (4)$$

Unlike Equation 3, the unlabeled instances are also considered in Equation 4 when finding the nearest neighbors. We can visualize this as an undirected graph, where each data instance (labeled or unlabeled) is a node that is connected to its k nearest neighbor nodes. The value of $y(\cdot)$ is set to 0 or 1 for labeled nodes depending on their class. For each unlabeled node x, $y(x)$ is equal to the average of the $y(\cdot)$ values of its neighbors. Such a function that satisfies the average property on all unlabeled nodes is called a *harmonic* function and is known to exist and have a unique solution (Doyle and Snell, 1984). Harmonic functions were first introduced as a semi-supervised learning method by Zhu *et al.* (2003). There are interesting alternative interpretations of a harmonic function on a graph. One of them can be explained in terms of random walks on a graph. Consider a random walk on a graph where at each time point we move from the current node to one of its neighbors. The next node is chosen among the neighbors of the current node with probability proportional to the weight (similarity) of the edge that connects the two nodes. Assuming we start the random walk from the node x, $y(x)$ in Equation 4 is then equal to the probability that this random walk will hit a node labeled 1 before it hits a node labeled 0.

[2] Equation 3 is the weighted (or *soft*) version of the kNN algorithm. In the classical *voting* scheme, x is classified in the category that the majority of its neighbors belong to.

4.2 Transductive SVM

Support vector machines (SVM) is a supervised machine learning approach designed for solving two-class pattern recognition problems. The aim is to find the decision surface that separates the positive and negative labeled training examples of a class with maximum margin (Burges, 1998).

Transductive support vector machines (TSVM) are an extension of SVM, where unlabeled data is used in addition to labeled data. The aim now is to assign labels to the unlabeled data and find a decision surface that separates the positive and negative instances of the original labeled data and the (now labeled) unlabeled data with maximum margin. Intuitively, the unlabeled data pushes the decision boundary away from the dense regions. However, unlike SVM, the optimization problem now is NP-hard (Zhu, 2005). Pointers to studies for approximation algorithms can be found in (Zhu, 2005).

In Section 3 we defined the similarity between two instances based on the cosine similarity and the edit distance based similarity between the paths in the instances. Here, we use these path similarity measures as kernels for SVM and TSVM and modify the SVM^{light} package (Joachims, 1999) by plugging in our two kernel functions.

A well-defined kernel function should be symmetric positive definite. While cosine kernel is well-defined, Cortes *et al.* (2004) proved that edit kernel is not always positive definite. However, it is possible to make the kernel matrix positive definite by adjusting the γ parameter, which is a positive real number. Li and Jiang (2005) applied the edit kernel to predict initiation sites in eucaryotic mRNAs and obtained improved results compared to polynomial kernel.

5 Experimental Results

5.1 Data Sets

One of the problems in the field of protein-protein interaction extraction is that different studies generally use different data sets and evaluation metrics. Thus, it is difficult to compare their results. Bunescu *et al.* (2005) manually developed the *AIMED* corpus[3] for protein-protein interaction and protein name recognition. They tagged 199 Medline abstracts, obtained from the Database of Interacting Proteins (DIP) (Xenarios et al., 2001) and known to contain protein interactions. This corpus is becoming a standard, as it has been used in the recent studies by (Bunescu et al., 2005; Bunescu and Mooney, 2005b; Bunescu and Mooney, 2006; Mitsumori et al., 2006; Yakushiji et al., 2005).

In our study we used the *AIMED* corpus and the *CB* (Christine Brun) corpus that is provided as a resource by BioCreAtIvE II (Critical Assessment for Information Extraction in Biology) challenge evaluation[4]. We pre-processed the CB corpus by first annotating the protein names in the corpus automatically and then, refining the annotation manually. As discussed in Section 3, we pre-processed both of the data sets as follows. We replicated each sentence for each different protein pair. For n different proteins in a sentence, $\binom{n}{2}$ new sentences are created, as there are that many different pairs of proteins. In each newly created sentence we marked the protein pair considered for interaction as *PROTX1* and *PROTX2*, and all the remaining proteins in the sentence as *PROTX0*. If a sentence describes an interaction between *PROTX1* and *PROTX2*, it is labeled as positive, otherwise it is labeled as negative. The summary of the data sets after pre-processing is displayed in Table 1[5].

Since previous studies that use AIMED corpus perform 10-fold cross-validation. We also performed 10-fold cross-validation in both data sets and report the average results over the runs.

[3]ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/

[4]http://biocreative.sourceforge.net/biocreative_2.html

[5]The pre-processed data sets are available at http://belobog.si.umich.edu/clair/biocreative

Data Set	Sentences	+ Sentences	- Sentences
AIMED	4026	951	3075
CB	4056	2202	1854

Table 1: Data Sets

5.2 Evaluation Metrics

We use precision, recall, and F-score as our metrics to evaluate the performances of the methods. Precision (π) and recall (ρ) are defined as follows:

$$\pi = \frac{TP}{TP+FP}; \qquad \rho = \frac{TP}{TP+FN} \tag{5}$$

Here, TP (True Positives) is the number of sentences classified correctly as positive; FP (False Positives) is the number of negative sentences that are classified as positive incorrectly by the classifier; and FN (False Negatives) is the number of positive sentences that are classified as negative incorrectly by the classifier.
F-score is the harmonic mean of recall and precision.

$$F\text{-}score = \frac{2\pi\rho}{\pi+\rho} \tag{6}$$

5.3 Results and Discussion

We evaluate and compare the performances of the semi-supervised machine learning approaches (TSVM and harmonic functions) with their supervised counterparts (SVM and kNN) for the task of protein-protein interaction extraction. As discussed in Section 3, we use cosine similarity and edit distance based similarity as similarity functions in harmonic functions and kNN, and as kernel functions in TSVM and SVM. Our instances consist of the shortest paths between the protein pairs in the dependency parse trees of the sentences. In our experiments, we tuned the γ parameter of the edit distance based path similarity function to 4.5 with cross-validation. The results in Table 2 and Table 3 are obtained with 10-fold cross-validation. We report the average results over the runs.

Table 2 shows the results obtained for the AIMED data set. Edit distance based path similarity function performs considerably better than the cosine similarity function with harmonic functions and kNN and usually slightly better with SVM and TSVM. We achieve our best F-score performance of 59.96% with TSVM with edit kernel. While SVM with edit kernel achieves the highest precision of 77.52%, it performs slightly worse than SVM with cosine kernel in terms of F-score measure. TSVM performs slightly better than SVM, both of which perform better than harmonic functions. kNN is the worst performing algorithm for this data set.

In Table 2, we also show the results obtained previously in the literature by using the same data set. Yakushiji *et al.* (2005) use an HPSG parser to produce predicate argument structures. They utilize these structures to automatically construct protein interaction extraction rules. Mitsumori *et al.* (2006) use SVM with the unparsed text around the protein names as features to extract protein interaction sentences. Here, we show their best result obtained by using the three words to the left and to the right of the proteins. The most closely related study to ours is that by Bunescu and Mooney (2005a). They define a kernel function based on the shortest path between two entities of a relationship in the dependency parse tree of a sentence (the SPK method). They apply this method to the domain of protein-protein interaction extraction in (Bunescu and Mooney, 2006). Here, they also test the methods ELCS (Extraction Using Longest Common Subsequences) (Bunescu et al., 2005) and SSK (Subsequence Kernel) (Bunescu and Mooney, 2005b). We cannot compare our results to theirs directly, because they report their results as a precision-recall graph. However, the best F-score in their graph seems to be around 0.50 and definitely lower than the best F-scores we have achieved (≈ 0.59). Bunescu and Mooney (2006) also use SVM as their learning method in their SPK approach. They define their similarity based on the number of overlapping words between two paths and assign a similarity of zero if the two paths have different lengths. Our improved performance with SVM and the shortest path dependency features may be due to the edit-distance based kernel, which takes into account not only the overlapping words, but also word order and accounts for deletions and insertions of words. Our results show that, SVM, TSVM, and harmonic functions achieve better F-score and recall performances than the previous studies by Yakushiji *et al.* (2005), Mitsumori *et al.* (2006), and the SSK and ELCS approaches of Bunescu and Mooney (2006). SVM and TSVM also achieve higher precision scores. Since, Mitsumori *et al.* (2006) also use SVM in their study, our improved results with SVM confirms our motivation of using dependency paths as features.

Table 3 shows the results we got with the CB data set. The F-score performance with the edit distance based similarity function is always better than that of cosine similarity function for this data set. The difference in performances is considerable for harmonic functions and kNN. Our best F-score is achieved with TSVM with edit kernel (85.22%). TSVM performs slightly better than SVM. When cosine similarity function is used, kNN performs better than harmonic functions. However, when edit distance based similarity is used, harmonic functions achieve better performance. SVM and TSVM perform better than harmonic functions. But, the gap in performance is low when edit distance based similarity is used with harmonic functions.

Method	Precision	Recall	F-Score
SVM-edit	77.52	43.51	55.61
SVM-cos	61.99	54.99	58.09
TSVM-edit	59.59	60.68	59.96
TSVM-cos	58.37	61.19	59.62
Harmonic-edit	44.17	74.20	55.29
Harmonic-cos	36.02	67.65	46.97
kNN-edit	68.77	42.17	52.20
kNN-cos	40.37	49.49	44.36
(Yakushiji et al., 2005)	33.70	33.10	33.40
(Mitsumori et al., 2006)	54.20	42.60	47.70

Table 2: Experimental Results – AIMED Data Set

Method	Precision	Recall	F-Score
SVM-edit	85.15	84.79	84.96
SVM-cos	87.83	81.45	84.49
TSVM-edit	85.62	84.89	85.22
TSVM-cos	85.67	84.31	84.96
Harmonic-edit	86.69	80.15	83.26
Harmonic-cos	72.28	70.91	71.56
kNN-edit	72.89	86.95	79.28
kNN-cos	65.42	89.49	75.54

Table 3: Experimental Results – CB Data Set

Semi-supervised approaches are usually more effective when there is less labeled data than unlabeled data, which is usually the case in real applications. To see the effect of semi-supervised approaches we perform experiments by varying the amount of la-

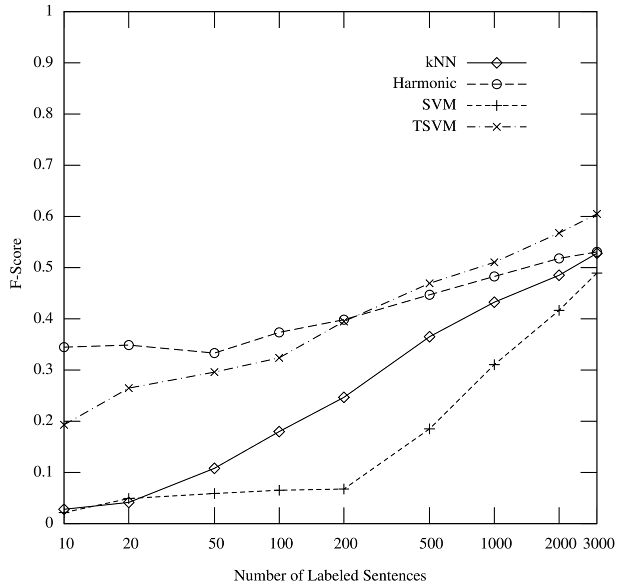

Figure 2: The F-score on the AIMED dataset with varying sizes of training data

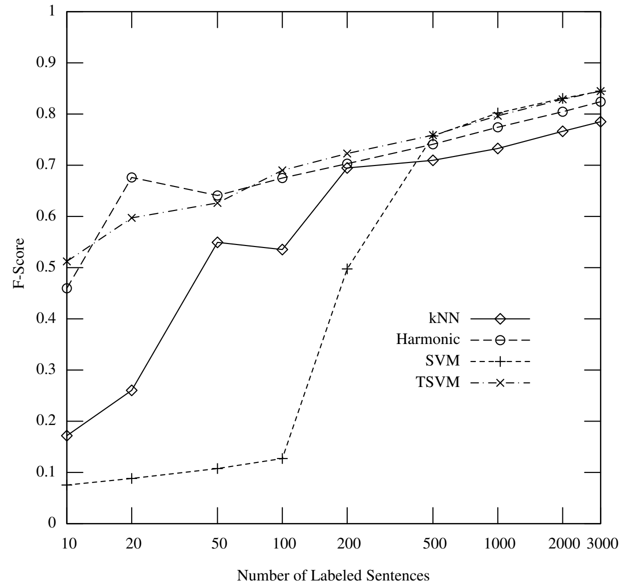

Figure 3: The F-score on the CB dataset with varying sizes of training data

beled training sentences in the range $[10, 3000]$. For each labeled training set size, sentences are selected randomly among all the sentences, and the remaining sentences are used as the unlabeled test set. The results that we report are the averages over 10 such random runs for each labeled training set size. We report the results for the algorithms when edit distance based similarity is used, as it mostly performs better than cosine similarity. Figure 2 shows the results obtained over the AIMED data set. Semi-supervised approaches TSVM and harmonic functions perform considerably better than their supervised counterparts SVM and kNN when we have small number of labeled training data. It is interesting to note that, although SVM is one of the best performing algorithms with more training data, it is the worst performing algorithm with small amount of labeled training sentences. Its performance starts to increase when number of training data is larger than 200. Eventually, its performance gets close to that of the other algorithms. Harmonic functions is the best performing algorithm when we have less than 200 labeled training data. TSVM achieves better performance when there are more than 500 labeled training sentences.

Figure 3 shows the results obtained over the CB data set. When we have less than 500 labeled sentences, harmonic functions and TSVM perform significantly better than kNN, while SVM is the worst performing algorithm. When we have more than 500 labeled training sentences, kNN is the worst performing algorithm, while the performance of SVM increases and gets similar to that of TSVM and slightly better than that of harmonic functions.

6 Conclusion

We introduced a relation extraction approach based on dependency parsing and machine learning to identify protein interaction sentences in biomedical text. Unlike syntactic parsing, dependency parsing captures the semantic predicate argument relationships between the entities in addition to the syntactic relationships. We extracted the shortest paths between protein pairs in the dependency parse trees of the sentences and defined similarity functions (kernels in SVM terminology) for these paths based on cosine similarity and edit distance. Supervised machine learning approaches have been applied to this domain. However, they rely only on labeled training data, which is difficult to gather. To our knowledge, this is the first effort in this domain to apply semi-supervised algorithms, which make use of both labeled and unlabeled data. We evaluated and compared the performances of two semi-supervised machine learning approaches (harmonic functions and TSVM), with their supervised counterparts (kNN and SVM). We showed that, edit distance based similarity function performs better than cosine similarity function since it takes into account not only common words, but also word order. Our 10-fold cross validation results showed that, TSVM performs slightly better than SVM, both of which perform better than harmonic functions. The worst performing algorithm is kNN. We compared our results with previous results published with the AIMED data set. We achieved the best F-score performance with TSVM with the edit distance kernel (59.96%) which is significantly higher than the previously reported results for the same data set.

In most real-world applications there are much more unlabeled data than labeled data. Semi-supervised approaches are usually more effective in these cases, because they make use of both the labeled and unlabeled instances when making decisions. To test this hypothesis for the application of extracting protein interaction sentences from text, we performed experiments by varying the number of labeled training sentences. Our results show that, semi-supervised algorithms perform considerably better than their supervised counterparts, when there are small number of labeled training sentences. An interesting result is that, in such cases SVM performs significantly worse than the other algorithms. Harmonic functions achieve the best performance when there are only a few labeled training sentences. As number of labeled training sentences increases the performance gap between supervised and semi-supervised algorithms decreases.

Acknowledgments

This work was supported in part by grants R01-LM008106 and U54-DA021519 from the US National Institutes of Health.

References

G. Bader, D. Betel, and C. Hogue. 2003. Bind - the biomolecular interaction network database. *Nucleic Acids Research*, 31(1):248–250.

A. Bairoch and R. Apweiler. 2000. The swiss-prot protein sequence database and its supplement trembl in 2000. *Nucleic Acids Research*, 28(1):45–48.

C. Blaschke, M. A. Andrade, C. A. Ouzounis, and A. Valencia. 1999. Automatic extraction of biological information from scientific text: Protein-protein interactions. In *Proceedings of the AAAI Conference on Intelligent Systems for Molecular Biology (ISMB 1999)*, pages 60–67.

R. C. Bunescu and R. J. Mooney. 2005a. A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731, Vancouver, B.C, October.

R. C. Bunescu and R. J. Mooney. 2005b. Subsequence kernels for relation extraction. In *Proceedings of the 19th Conference on Neural Information Processing Systems (NIPS)*, Vancouver, B.C, December.

R. C. Bunescu and R. J. Mooney, 2006. *Text Mining and Natural Language Processing*, chapter Extracting Relations from Text: From Word Sequences to Dependency Paths. forthcoming book.

R. Bunescu, R. Ge, J. R. Kate, M. E. Marcotte, R. J. Mooney, K. A. Ramani, and W. Y. Wong. 2005. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155, February.

C. J. C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

C. Cortes, P. Haffner, and M. Mohri. 2004. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research*, (5):1035–1062, August.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, Barcelona, Spain, July.

N. Daraselia, A. Yuryev, S. Egorov, S. Novichkova, A. Nikitin, and I. Mazo. 2004. Extracting human protein interactions from medline using a full-sentence parser. *Bioinformatics*, 20(5):604–611.

M-C. de Marneffe, B. MacCartney, and C. D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the IEEE / ACL 2006 Workshop on Spoken Language Technology*. The Stanford Natural Language Processing Group.

I. Donaldson, J. Martin, B. de Bruijn, C. Wolting, V. Lay, B. Tuekam, S. Zhang, B. Baskin, G. D. Bader, K. Michalockova, T. Pawson, and C. W. V. Hogue. 2003. Prebind and textomy - mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics*, 4:11.

P. G. Doyle and J. L. Snell. 1984. *Random Walks and Electric Networks*. Mathematical Association of America.

M. Huang, X. Zhu, Y. Hao, D. G. Payan, K. Qu, and M. Li. 2004. Discovering patterns to extract protein-protein interactions from full texts. *Bioinformatics*, 20(18):3604–3612.

T. Joachims. 1999. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann Publishers, San Francisco, US.

H. Li and T. Jiang. 2005. A class of edit kernels for svms to predict translation initiation sites in eukaryotic mrnas. *Journal of Computational Biology*, 12(6):702–718.

T. Mitsumori, M. Murata, Y. Fukuda, K. Doi, and H. Doi. 2006. Extracting protein-protein interaction information from biomedical text with svm. *IEICE Transactions on Information and Systems*, E89-D(8):2464–2466.

T. Ono, H. Hishigaki, A. Tanigami, and T. Takagi. 2001. Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics*, 17(2):155–161.

E. M. Phizicky and S. Fields. 1995. Protein-protein interactions: methods for detection and analysis. *Microbiol. Rev.*, 59(1):94–123, March.

J. Pustejovsky, J. Castano, J. Zhang, M. Kotecki, and B. Cochran. 2002. Robust relational parsing over biomedical literature: Extracting inhibit relations. In *Proceedings of the seventh Pacific Symposium on Biocomputing (PSB 2002)*, pages 362–373.

K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. 2003. Extracting information on protein-protein interactions from biological literature based on machine learning approaches. *Genome Informatics*, 14:699–700.

J. M. Temkin and M. R. Gilder. 2003. Extraction of protein interaction information from unstructured text using a context-free grammar. *Bioinformatics*, 19:2046–2053.

I. Xenarios, E. Fernandez, L. Salwinski, X. J. Duan, M. J. Thompson, E. M. Marcotte, and D. Eisenberg. 2001. Dip: The database of interacting proteins: 2001 update. *Nucleic Acids Res.*, 29:239 – 241, January.

A. Yakushiji, Y. Miyao, Y. Tateisi, and J. Tsujii. 2005. Biomedical information extraction with predicate-argument structure patterns. In *Proceedings of The Eleventh Annual Meeting of The Association for Natural Language Processing*, pages 93–96.

A. Zanzoni, L. Montecchi-Palazzi, M. Quondam, G. Ausiello, M. Helmer-Citterich, and G. Cesareni. 2002. Mint: A molecular interaction database. *FEBS Letters*, 513:135–140.

X. Zhu, Z. Ghahramani, and J. D. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In T. Fawcett and N. Mishra, editors, *ICML*, pages 912–919. AAAI Press.

X. Zhu. 2005. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison. http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf.