Extracting Interacting Protein Pairs and Evidence Sentences by using Dependency Parsing and Machine Learning Techniques

Güneş Erkan[1]   **Arzucan Özgür**[1]   **Dragomir R. Radev**[1,2]

gerkan@umich.edu   ozgur@umich.edu   radev@umich.edu

[1]   Department of Electrical Engineering and Computer Science, University of Michigan,
       Ann Arbor, MI 48109, USA
[2]   School of Information, University of Michigan, Ann Arbor, MI 48109, USA

Abstract

The biomedical literature is growing rapidly. This increases the need for developing text mining techniques to automatically extract biologically important information such as protein-protein interactions from free texts. Besides identifying an interaction and the interacting pair of proteins, it is also important to extract from the full text the most relevant sentences describing that interaction. These issues were addressed in the BioCreAtIvE II (Critical Assessment for Information Extraction in Biology) challenge evaluation as sub-tasks under the protein-protein interaction extraction (PPI) task. We present our approach of using dependency parsing and machine learning techniques to identify interacting protein pairs from full text articles (Protein Interaction Pairs Sub-task 2 (IPS)) and extracting the most relevant sentences that describe their interaction (Protein Interaction Sentences Sub-task 3 (ISS)).

1   Introduction

Protein-protein interactions play important roles in vital processes such as cell cycle control, and metabolic and signaling pathways. There are a number of (mostly manually curated) databases such as MINT [11] and SwissProt [1] that 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.

There have been many approaches to extract protein interactions from free texts. One approach is matching pre-specified patterns and rules [2]. Although this approach achieves high precision, it suffers from low recall. The reason is that, cases which are not covered by the pre-defined patterns and rules can not be extracted. Another approach is using natural language processing (NLP) techniques such as full parsing [4] and partial parsing [8]. These parsing approaches consider sentence syntax only but not its semantics. 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 [5, 7, 10]. These studies usually use bag-of-words features, or only syntactic features extracted from sentences and do not consider any dependency or semantic information.

BioCreAtIvE II (Critical Assessment for Information Extraction in Biology) challenge evaluation [1] consists of three tasks, which are Gene Mention Tagging (GM), Gene Normalization (GN), and Protein Protein Interaction (PPI). We participated in two sub-tasks of PPI, Protein Interaction Pairs

---

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

Sub-task 2 (IPS) and Protein Interaction Sentences Sub-task 3 (ISS). In these subtasks participants were provided with a collection of 358 full text articles in HTML. The aim of IPS was to identify the interacting protein pairs in each article. The goal of ISS was to extract the most relevant sentences that describe an interaction between two given proteins. For each protein interaction pair, participants were required to return a ranked list of maximum 5 evidence passages describing their interaction. Here, we present our approach of using dependency parsing and machine learning techniques to handle these tasks. We extract features from the dependency parse trees of the sentences and use these features to train an SVM classifier to identify and rank sentences that describe an interaction. Dependency parse trees not only capture sentence syntax but also some of its semantics such as predicate-argument relationships. We also present the improved version of our system, where we extract paths between a protein pair in the dependency parse tree of a sentence and define two kernel functions for SVM based on the cosine and edit distance based similarities among these paths.

2   System Description

2.1   Pre-processing

In this step, the HTML articles are converted to plain text by using html2text tool [2]. Next, tokenization is done such that each alphanumeric word and punctuation mark is considered as a separate token. Finally, articles are segmented into sentences by using the MxTerminator tool [9].

2.2   Protein Name Identification

In order to extract protein-protein interaction information from an article, first the protein names must be identified. For the BioCreAtIvE II challenge we were provided with a release of the SwissProt database [1]. We adapted the dictionary-based approach to identify protein names. We used the provided database as a dictionary to match the words in a sentence against the "description", "gene name", and "gene synonyms" fields. We preferred longer matches to shorter ones. If a sentence contains n different proteins, there are $\binom{n}{2}$ different pairs of proteins. 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*.

2.3   Protein Name Conflict Resolution

For the BioCreAtIvE challenge we were supposed to output the UniProt IDs of the protein pairs that interact, not their names. Since proteins with the same name may have different UniProt IDs depending on their organism source, we use heuristics to identify the organism source of a protein and map a protein name to its corresponding UniProt ID. For each protein, we matched the candidate organism names and synonyms in the article and weighted them according to their proximity to the protein. In our weighting mechanism, the frequencies of the organism name appearing just before the protein name, in the same sentence with the protein name, and in the same article with the protein name are considered in descending order of importance. For instance, suppose we have a protein name "Alpha-adaptin A". This protein has two candidate organism sources (human and mouse) and thus two candidate UniProt IDs (AP2A1_HUMAN and AP2A1_MOUSE) according to the UniProt table. We apply the following rule to select the correct UniProt ID of this protein.

1. Select the organism source that has the highest frequency of occurrence just before the protein name.
2. If it can not be disambiguated by (1), then select the organism source that has the highest frequency of occurrence in the same sentence with the protein name
3. If it can not be disambiguated by (1) and (2), then select the organism source that has the highest frequency of occurrence in the same article with the protein name

---

[2]http://userpage.fu-berlin.de/ mbayer/tools/html2text.html

4. If it can not be disambiguated by (1), (2), and (3), then select the organism source human (if one of the candidates is human)
5. If it can not be disambiguated by (1), (2), (3), and (4), then the conflict can not be resolved.

2.4 Sentence Filtering

We assumed that protein-protein interaction sentences contain at least two proteins and an interaction word. Thus, we consider only such sentences and filter all the others. A list of interaction words, which consists of 45 noun and 53 verb roots, was compiled from the literature. We extended the list to contain all the inflected forms of the words and spelling variations such as *coactivate/co-activate* and *localize/localise*.

2.5 Feature Extraction with Dependency Parsing

Unlike constituent parsing, dependency parsing captures the semantic predicate-argument relationships in a sentence. We used the Stanford Parser [3] to extract features from the dependency trees. 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." The final list of features used in our learning-based system is as follows.

- Each interaction word in our list is a binary feature by itself. In other words, if a particular interaction word occurs in a sentence, we set the corresponding feature for that sentence. If an interaction word is negated in the dependency tree, then we do not include that word, i.e. we assume that it does not occur in the sentence.
- A binary feature that is set if the total distance of both protein names to an interaction verb in a sentence is 2, that is, if both protein names are the immediate children of an interaction verb.
- An interaction verb that is an immediate parent of both proteins in the tree is a feature by itself.
- The immediate parent node of each protein in the dependency tree is a feature by itself.
- An interaction word that is an ancestor of a protein at one or two levels above it in the dependency tree is a feature by itself.
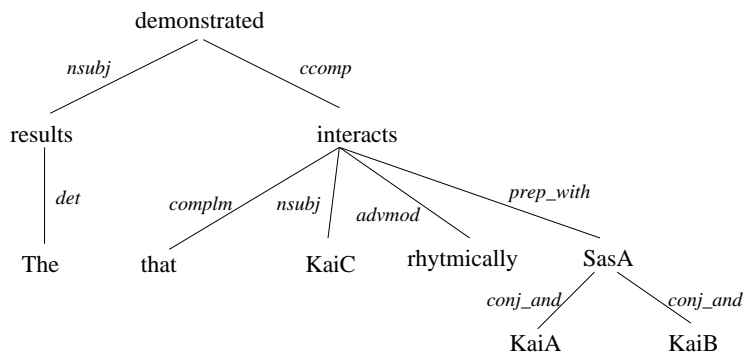

Figure 1: The dependency tree of the sentence "*The results demonstrated that KaiC interacts rhythmically with KaiA, KaiB, and SasA.*"

2.6 Machine Learning Techniques for Sentence Classification and Ranking

To classify sentences as containing an interaction or not and to rank the interaction sentences for each pair of proteins we used support vector machines (SVM). It has been shown to be one of the most powerful classifiers in general as well as the biomedical domain [5, 7, 10]. Our task is slightly different from that of the previous studies. Besides identifying protein-protein interaction pairs and sentences, we also identify the best sentences describing an interaction of a specific protein pair. We employ the strengths of both SVM and dependency parsing. We use the SVM^{light} library with linear kernel and default parameters [6] to classify sentences as containing an interaction or not and to rank the sentences that are classified as containing an interaction for each pair of interacting proteins.

---

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

As training set we used the Christine Brun corpus, provided as a resource by BioCreAtIvE. We first annotated the protein names in the corpus automatically and then, refined the annotation manually. As discussed in Section 2.2, each protein pair is marked as PROTX1 and PROTX2; and the remaining proteins in the sentence are marked as PROTX0. We ended up with $4,056$ sentences. $2,704$ of them are used for training and $1,352$ for test. In the end, the system is trained with all the $4,056$ sentences.

We ran the trained classifier on the test sentences and got a score (positive or negative) for each candidate sentence. To decide on whether we should output any sentence for a protein pair, we add up all the scores of the candidate sentences that contained the particular pair. If the sum for a pair is above a threshold, we output the top scoring (maximum of five) sentences for that pair and also output that pair as interacting. The threshold is set to 0 and 1 in runs 1 and 2, respectively. In run 3, we output the protein pair as containing an interaction, if there is at least one sentence that scored positive (IPS sub-task) and output the top 5 positive sentences for that pair (ISS sub-task).

2.7   Mapping Text to HTML

A requirement of the BioCreAtIvE challenge was that the predicted interaction sentences should come from the full text of the test set HTML articles. So, we had to map the extracted text sentences back to their HTML counterparts. We implemented an approximate string matching algorithm based on Levenshtein (edit) distance and an approximate token matching algorithm to handle this problem. First, we extracted the html passage where the sentence appears by using the approximate string matching algorithm. Next, we extracted the exact html sentence from that passage with the approximate token matching algorithm.

3   Improved System

Here, we present our improved system, where we use the shortest paths between a protein pair in the dependency parse tree of the sentence as features. We define two kernel functions for SVM based on the similarity between these paths. This system is developed after the submissions to the BioCreAtIvE challenge. Thus, all of our submitted runs used the system described in Section 2.

3.1   Sentence Similarity Based on Dependency Parsing

We define the similarity between two sentences based on the paths between two proteins in the dependency parse trees of the sentences. From the dependency parse trees of each sentence we extract the shortest path between a protein pair. For instance, in Figure 1 the path between $KaiC$ and $SasA$ is "KaiC - nsubj - interacts - prep_with - SasA". Since, this sentence defines an interaction between $KaiC$ and $SasA$, this is a positive instance. The path between $SasA$ and $KaiA$ is "SasA - conj_and - KaiA". This sentence does not describe an interaction between $SasA$ and $KaiA$. Thus, this path is a negative instance. If more than one path exists between the two proteins in a pair in the sentence (this may be the case if either of the proteins occurs more than once in the sentence), we select the shortest path. In our example sentence, there is a single path between each pair of proteins.

We define the similarity between two instances using cosine similarity and edit distance based similarity between the paths in the instances as follows.

3.1.1   Cosine Similarity

Suppose p_i and p_j are the paths between a protein pair 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:

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

3.1.2 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. 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 $[0, 1]$.

3.2 Kernel Function Definitions for SVM

We introduce two kernel functions for SVM based on the similarity functions that we defined in the previous sub-section. The cosine similarity based kernel K_{cos} and the edit distance based similarity kernel K_{edit} are defined as follows:

$$K_{cos}(x_i, x_j) = cos\_sim(x_i, x_j); \quad K_{edit}(x_i, x_j) = e^{-\gamma \times edit\_distance(x_1, x_2)} \tag{2}$$

The parameter γ is a positive number that allows us to tune K_{edit} to be symmetric positive definite, i.e., a well-defined kernel function.

4 Results and Discussion

Table 1 shows the summary of our results compared to the results obtained by the median system in the IPS sub-task. Interaction Pairs is the performance on identifying the interacting protein pairs. Interactor Normalization is the performance on interactor protein - article associations. To compute the average scores (av.) the scores for each article are computed separately and then the average is takes. However, some articles contain a single interaction, while other contain many interactions. Thus, the overall scores are also presented. Here, we report the scores of our best run (Run 2), in terms of F-score in the Interaction Pairs - Overall evaluation. We report the results for the cases where, articles containing exclusively interaction pairs that can be normalized to SwissProt entries are considered. Our results are higher than the median system in terms of all the evaluation metrics. Table 2 shows the summary results of our best run (Run 3), in terms of MRR score compared to the average results of all participating teams in the ISS sub-task. # Pred (A) is number of all predicted passages, # Pred (U) is number of unique passages, # TP (A) is number of and % (A) is fraction of true positives out of all predictions, # TP (U) is number of and % (U) fraction of true positives out of unique predictions, MRR is mean reciprocal rank of correct unique passages. The number of passages that we have predicted is higher than the average number of passages predicted by all the teams. Although the number of our true positives is much greater than the average, when we take the fraction over all the predictions it drops down. Our MRR score is slightly below the average MRR score reported by the BioCreative committee. However, they discuss that this score is not really statistically meaningful, as some teams didn't use the ranking system defined for this task. In their runs, each team was required to submit a ranked list of maximum 5 evidence passages for each interacting protein pair in each article. However, some teams ranked all the submitted passages as 1 in their runs, and some performed the ranking not for each pair but for the whole article. The BioCreative committee state that, when such runs are excluded from the statistics, the average MRR drops.

There is a lot of room for improvement in our system for both the IPS and ISS tasks. In the previous section, we discussed an improved version of our system, where we define kernel functions for

SVM based on the edit distance and cosine similarity among the paths between a protein pair in the dependency parse trees of the sentences. Our experiments with the Christine Brun corpus, show this system performs better in classifying a protein pair as interacting or not. However, all the other steps in the pipeline such as protein name identification, source organism disambiguation, and mapping text sentences back to html have an important affect on the overall performance and can be improved considerably.

Table 1: Summary of results of the IPS sub-task

Evaluation	Precision	Recall	F-score
Interaction Pairs - Overall (our results)	0.0759	0.1285	0.0954
Interaction Pairs - Overall (median)	0.0649	0.1179	0.0769
Interaction Pairs - Av. (our results)	0.0940	0.1988	0.0978
Interaction Pairs - Av. (median)	0.0808	0.2156	0.0842
Interactor Normalization - Overall (our results)	0.1478	0.3036	0.1988
Interactor Normalization - Overall (median)	0.1337	0.2723	0.1683
Interactor Normalization - Av. (our results)	0.2122	0.3269	0.2331
Interactor Normalization - Av. (median)	0.1707	0.3060	0.1922

Table 2: Summary of results of the ISS sub-task

Team	# Pred (A)	# TP (A)	# Pred (U)	# TP (U)	% (A)	% (U)	MRR
Our results	8355	5172	290	163	0.0347	0.0315	0.5329
Av. of all teams	6213.53846	3429.65385	207.46154	128.61538	0.04727	0.04725	0.55737

References

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

[2] Blaschke, C., Andrade, M. A., Ouzounis, C. A., and Valencia, A. (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.

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

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

[5] Donaldson, I., Martin, J., de Bruijn, B., Wolting, C., Lay, V., Tuekam, B., Zhang, S., Baskin, B., Bader, G. D., Michalockova, K., Pawson, T., and Hogue, C. W. V. (2003). Prebind and textomy - mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics*, **4**, 11. / *Bioinformatics*, **17**(Suppl 1), S97–S106.

[6] Joachims, T. (1999). *Advances in Kernel Methods-Support Vector Learning*, chapter Making Large-Scale SVM Learning Practical. MIT-Press.

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

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

[9] Reynar, J. C. and Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, D.C, USA.

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

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