Identifying Protein-Protein Interaction Sentences Using Boosting and Kernel Methods

Soo-Yong Shin[13] syshin@nist.gov
Sun Kim[23] skim@bi.snu.ac.kr
Jae-Hong Eom[2] jheom@bi.snu.ac.kr
Byoung-Tak Zhang[2] btzhang@bi.snu.ac.kr
Ram Sriram[1] sriram@nist.gov

[1] Manufacturing Systems Integration Division, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA
[2] Biointelligence Laboratory, School of Computer Science and Engineering, Seoul National University, Seoul 151-744, Korea
[3] Both authors have equally contributed to this work.

Abstract

As the amount of biological research literature increases, finding information is becoming a daunting task. Since machine learning techniques could alleviate this problem, we propose a machine learning framework to identify protein-protein interaction sentences from research papers. This machine learning technique is one of the basic components needed to automatically extract biological information from texts. Since the protein-protein interaction (PPI) sentences have their own patterns at article and sentence levels, these patterns are mined by using boosting and kernel methods. Both approaches have good characteristics for the PPI extraction tasks, and naturally can handle heuristic information for future extensions.

Keywords: Protein-Protein Interaction Identification, Boosting Methods, Tree Kernels, Support Vector Machines

1 Introduction

The growing accumulation of functional descriptions in biomedical literature necessitate the importance of text mining tools to facilitate the extraction of such information [1]. Therefore, diverse approaches such as pattern matching, statistical learning, and natural language processing have been proposed. Here, we present a machine learning-based framework, in particular, without any prior knowledge other than training data. In biological text mining, only a small amount of annotated documents are available for public use, which limits the usage of machine learning techniques. Nevertheless, it is important to examine the ability of machine learning methods to determine the possibility for real-world use, because the heuristic approaches (with or without learning) might be ineffective in the long term.

The goal of the BioCreative project is to evaluate text mining and information extraction systems applied to the biological domain [1]. We participated in two subtasks of the Protein-Protein Interaction (PPI) task in the BioCreative II competition. The subtasks of PPI of interest to us are the Protein Interaction Article (IAS) subtask and the Protein Interaction Sentence (ISS) subtask. The IAS subtask is the classification of whether a given article contains protein interaction information. It is the first step to extract the PPI information, by selecting those articles which have relevant information related to protein interactions. The IAS system should return a ranked list of PPI articles based on their relevance in the task. Before getting protein interaction pairs, it is useful to select the most relevant sentences which are directly connected to the protein interactions. The ISS subtask is to filter

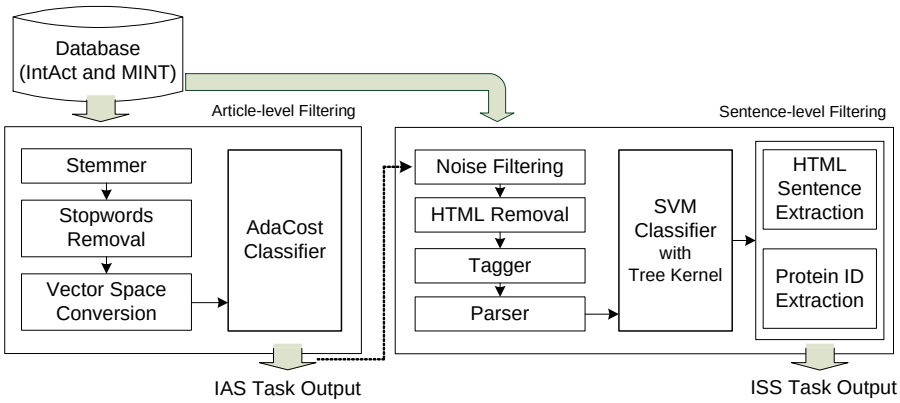

Figure 1: Overview of the PPI extraction system for BioCreative II. Dotted line is not implemented and not used for the BioCreative tasks as the committee provides two separate training sets.

those PPI relevant sentences. The ISS system is required to submit a ranked list of HTML passages describing protein-protein interactions.

For the IAS subtask, the AdaCost [5], a cost-sensitive learning algorithm, is used to give bias towards PPI relevant documents. Since we use naive Bayes classifiers as weak learners in the AdaCost framework, any prior knowledge can be naturally adapted in probabilistic form. For the ISS subtask, a tree kernel method [4] is utilized to mine the PPI patterns among sentences, which is based on the assumption that the PPI information tends to be written in specific grammatical structure [6]. It also can employ additional heuristic knowledge in an easy way.

The paper is organized as follows: In Section 2, the proposed PPI extraction approaches are described and analyzed. Concluding remarks and future research are provided in Section 3.

2 Methods and Analysis

The proposed PPI extraction system consists of two parts: 1) article-level and 2) sentence-level filters. These filters are for the IAS subtask and the ISS subtask, respectively. Figure 1 shows the overview of the two-phase PPI extraction. Free texts enter the article-level filter at first, which identifies the PPI relevant articles using the AdaCost classifier. After the PPI articles are classified, the PPI information is picked up at the sentence level. The second phase uses support vector machines (SVMs) with tree kernels. Although the article-level and the sentence-level filters are combined together as a complete PPI extraction system, the two phases are separately performed for the BioCreative tasks, and each produces its own result according to the participating subtasks.

2.1 PPI Article Filtering by Cost-Sensitive Learning

The IAS subtask is the first step to extract the PPI information at article level, so that the actual extractor (ISS system) can use less-noisy data. At this point, the filtering system should not miss any PPI relevant document even though a certain amount of irrelevant documents are included in the filtered set, i.e., recall is more important than precision. To handle the tradeoff between recall and precision, our system utilizes a cost-sensitive learning algorithm, AdaCost [5]. Unlike other machine learning classifiers, which focus on minimizing the number of incorrect predictions, AdaCost provides the flexibility between precision and recall rates by using a cost factor. It is similar to AdaBoost [8], but the main difference is how the data distribution is updated. AdaCost has an additional parameter, so-called "cost" in updating the data distribution. The weight of an instance with high cost will be changed greater than the weight of an instance with low cost. This allows the learning system to

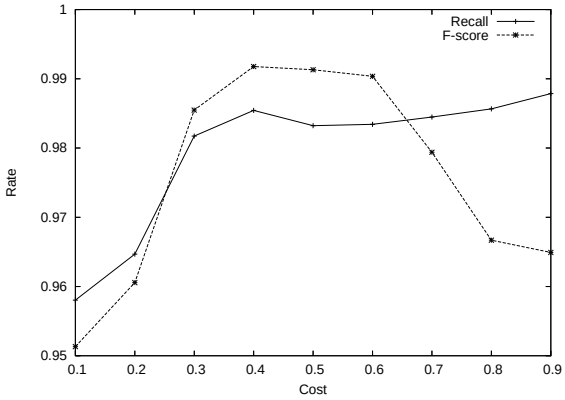

Figure 2: Recall and F-score changes for cost on unbalanced dataset.

classify high-cost instances more correctly. We use naive Bayes learning as weak learner which is known to be efficient in text filtering [7]. In addition, the naive Bayes classifier is suitable for our purpose of participating in the BioCreative II, which is to build a machine learning framework that can be further used to adapt heuristic knowledge in easy ways. The naive Bayes classifier is a statistical learning method that can naturally use the heuristic knowledge only if it can be transformed into probabilities. The modified AdaCost with naive Bayes algorithm used for the article-level filtering is as follows (our modification is shown in bold letters):

- Given training examples $S = \{(x_1, c_1, y_1), \ldots, (x_m, c_m, y_m)\}$;
  x_i is instance ($x_i \in X$), c_i is cost factor ($c_i \in R^+$), and y_i is label ($y_i \in \{-1, +1\}$).
- Initialize $D_1(i)$ (such as $D_1(i) = c_i / \sum_j^m c_j$).
- For $t = 1, \ldots, T$:
  1. **Train a naive Bayes classifier using distribution D_t.**
  2. Compute weak hypothesis $h_t : X \rightarrow R$.
  3. Choose $\alpha_t \in R$ and $\beta(i) \in R^+$,
     where α_t is a weight parameter for weak hypothesis h_t at the t-th round, and $\beta(i) = \beta\left(\text{sign}(y_i h_t(x_i)), c_i\right)$ is a cost-adjustment function.
  4. Update $D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i) \beta(i))}{Z_t}$, where Z_t is a normalization factor.
- Output the final hypothesis:
  $H(x) = \text{sign}(f(x))$ where $f(x) = \left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

The training data used in the IAS subtask contains 3,536 positive examples and 1,959 negative examples. The noisy positive examples given by the committee are excluded from the experiments. For the AdaCost classifier, the documents are preprocessed by stemming and stopword removal [7]. We use a modified stopword list, where the PPI-related words are omitted from common stopwords. Then the remaining texts are converted to the bag-of-words representation because we presume that some specific words or the simple combination of the words are enough to evaluate the PPI relevance of the articles.

Figure 2 presents recall and F-score changes for the cost c_i on training data. The overall best performance occurs at 0.4 cost, whereas the highest recall is achieved at 0.9 cost. The unusual peak of 0.4 cost is caused by the unbalanced number of positive and negative examples and relatively small size of dataset. In the article-level filtering, the recall is more important unless the F-score drops drastically, hence higher cost is preferred. However, for the official run of the IAS subtask, the cost

was set to 0.5 since it is an independent subtask from other PPI subtasks, and only evaluated by the IAS system output. Our IAS system got 65.73% of accuracy and 71.54% of F-score on test data. We found out that there is a different PPI-related vocabulary between training examples and test examples, which bears the performance decrease on test data. This problem can be solved by using PPI-related dictionaries or databases, which remains as future research.

2.2 PPI Sentence Filtering by Tree Kernels

The ISS subtask consists of two steps: choosing relevant sentences and finding UniProt IDs of interacting protein pair. For the first step, we assume the PPI sentences can be discriminated by investigating their grammatical structures, since most of PPI sentences tend to have unique grammatical structures [6]. A parsing tree in natural language processing represents a set of words and its structural information. The convolution kernel was chosen to calculate structural similarity among parsing trees [4].

In the convolution tree kernel (Φ) algorithm, kernel value is evaluated by summing up the number of common subtrees between two trees to calculate the structural similarity. A tree is represented as a vector of subtrees through high dimensional feature mapping [4]:

$$\Phi(\text{Tree } T) = (subTree(\text{type } 1), \ldots, subTtree(\text{type } n)),$$

where $subTree(\text{type } n)$ is the number of subtree of node type n. Then, the kernel function is defined as follows:

$$K(T_1, T_2) = \langle \Phi(T_1) \cdot \Phi(T_2) \rangle = \sum_l \Phi(T_1)[i] \times \Phi(T_2)[i] = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1) \times I_i(n_2),$$

where N_1 and N_2 represent the set of all possible nodes of trees T_1 and T_2, and $I_i(n)$ is an indicator function which has 1 if sub-tree of type i starts from root node n, 0 otherwise.

The number of subtrees with type i in tree T is calculated by $\Phi(T)[i] = \sum_{n \in N} I_i(n)$, which gives the total number of nodes in tree T which have subtrees with type i. The inner product between two trees, having its features as the all possible subtrees, is computed by the following recursive way and it is known to be calculated in polynomial time.

- If the form of the child nodes of n_1 and n_2 are different, $NCS(n_1, n_2) = 0$, where $NCS(n_1, n_2)$ is the number of common subtree between n_1 and n_2.
- If the form of the child nodes of n_1 and n_2 are identical (including their order) and they are leaf nodes, $NCS(n_1, n_2) = \lambda$.
- For all other cases, $NCS(n_1, n_2) = \prod_j (1 + NCS(ch(n_1)_j, ch(n_2)_j))$, where $ch(n_1)_j$ is the j-th child of node n_1, $ch(n_2)_j$ is the j-th child of node n_2, and $NCS(ch(n_1)_j, ch(n_2)_j) = \lambda \sum_i I_i(n_1) \times I_i(n_2)$. The parameter λ, $0 < \lambda \leq 1$, is used to consider the relative importance of tree fragment according to its length and is set to '1' when the size of tree fragments is not considered.

To achieve the parsing tree of the sentence, we use the following procedure. First, we extract plain texts by removing HTML tags in HTML documents to use the grammatical structure information. Second, the extracted sentences are tagged by a rule-based part-of-speech tagger [2]. The Brill tagger is trained beforehand, using GENIA corpus (available at `http://www-tsujii.is.s.u-tokyo.ac.jp/~genia/topics/Corpus`). Third, the tagged sentences are parsed by a statistical natural language parser [3]. Then, the irrelevant parsing trees such as a noun phrase are discarded, since they do not contain the meaningful grammatical structure. This leads to some positive examples that only have noun phrases be excluded from training data. After calculating the kernel tree, the interaction

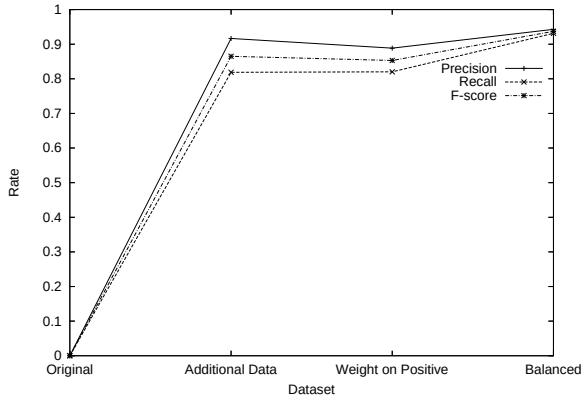

Figure 3: Precision, recall and F-score changes for training data sets.

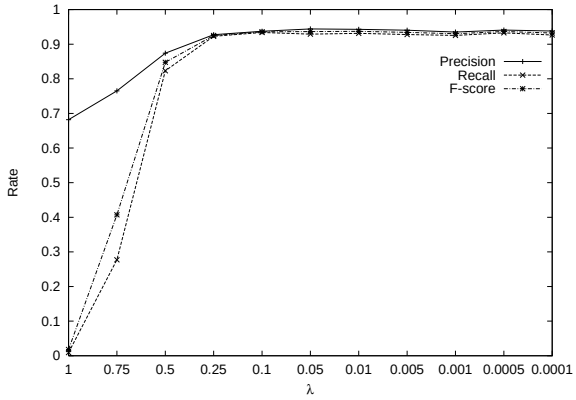

Figure 4: The effect of λ in tree kernel calculated with balanced dataset.

patterns are learned by support vector machines (SVM). We use the LIBSVM package (available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`) which can handle pre-computed kernel matrices.

Training data for the ISS subtask consists of 1,634 positive sentences and 1,763 negative sentences. We try to balance the positive and negative examples to improve the quality because the excessive negative examples in the original dataset force the SVM classifier to turn all test sentences into negative examples. We incorporate the Anne-Lise-Veuthey corpus and the PRODISEN Interaction corpus to enrich the positive examples, which are also released by the committee for the ISS subtask. For the official run, we use about 10% more negative examples than positive ones, so that we give a slight bias to non-relevant PPIs, and can get reduced computational time. Note that only a few sentences are available as positive examples at sentence-level filtering out of whole texts.

Figure 3 shows the performance changes for 4 different training data sets. The results were obtained from 10-fold cross-validation. The "Original" means the first standard dataset provided by the ISS subtask. The "Additional Data" is created by adding the corpus, Anne-Lise-Veuthey and PRODISEN Interaction, and the second standard dataset to the "Original." The "Weights on Positive" gives more weight to positive examples in the "Additional Data." The "Balanced" is the balanced dataset, where the number of negative examples is only 10% more than that of positive examples. The balanced dataset gains the best performance, and it shows the importance of making balances between positive and negative examples. The effect of λ in the tree kernels was also examined. Figure 4 shows the experimental results. Since sentence lengths are very diverse, λ should be carefully chosen. The best performance is taken when λ is 0.01, and we got 94.30% precision, 93.15% recall, and 93.72% F-score on the balanced training data. According to the preliminary results, we found that the tree kernel provides good predictions if the corpus is limited to certain conditions for both training and test data.

In the submitted run of the ISS subtask, we used the reduced sentences which removed the words tagged by less important elements such as articles, adverbs, and adjectives to save computational time. However, the follow-up experiments showed that using original sentences provides better performance for all criteria. Because the answers for the ISS test data have not been published yet, we could not analyze the proposed method and its variants further.

Even though we concentrated on the HTML sentence extraction, we also implemented the protein ID extraction module using a simple word-to-word matching approach to find protein IDs from the selected PPI sentences. A UniProt ID dictionary is built with gene names, aliases, orf names, and protein descriptions. Simple morphological variations for each protein term are considered to increase the coverage in the searching process. We also consider compound words by using bi-gram and tri-gram of a sentence. Finally, the nearest two UniProt IDs found in a sentence are selected as a system result.

3 Summary

We presented a machine learning approach to extract protein-protein interactions. This method consists of two procedures: article-level filtering and sentence-level filtering. In the article-level filtering, documents are roughly classified to reduce the overhead in the second procedure. The AdaCost with naive Bayes classifiers is used for the article-level filtering, and the SVM classifier with tree kernels is used to identify PPI relevant sentences as sentence-level filtering.

Our focus is to develop a machine learning-based framework, which can be further enhanced by adding heuristic techniques because it extends the system performance particularly in the biomedical domain. In the present work, we did not apply any heuristic approaches such as protein/interaction word dictionaries. Previous research indicates that the dictionary method could increase the PPI extraction performance when the training data size is limited. Thus, study on exploring efficient heuristic approaches remains as a future research work.

Acknowledgments

This work was supported by the Korea Science and Engineering Foundation(KOSEF) through the National Research Lab. Program funded by the Ministry of Science and Technology (No. M10400000349-06J0000-34910). Soo-Yong Shin was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2006-214-D00140) and the Manufacturing Metrology and Standards for the Health Care Enterprise Program at NIST. Jae-Hong Eom was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2006-511-D00355). Mention of commercial products or services in this paper does not imply approval or endorsement by NIST, nor does it imply that such products or services are necessarily the best available for the purpose.

References

[1] C. Blaschke, E. A. Leon, M. Krallinger, and A. Valencia, Evaluation of BioCreAtIvE Assessment of Task 2, *BMC Bioinformatics*, 6(Suppl 1):S16, 2005.

[2] Brill, E., A Simple Rule-Based Part-Of-Speech Tagger, *Proc. 3rd Conf. on Applied Natural Language Processing*, 152–155, 1992.

[3] Collins, M., Head-Driven Statistical Models for Natural Language Parsing, *PhD Dissertation*, University of Pennsylvania, 1999.

[4] Collins, M. and Duffy, N., Convolution Kernels for Natural Languages, *Proc. 15th Conf. on Neural Information Processing Systems*, 625–632, 2001.

[5] Fan, W., Stolfo, S., Zhang, J., and Chan, P., AdaCost: Misclassification Cost-Sensitive Boosting, *Proc. 16th Inter. Conf. on Machine Learning*, 97–105, 1999.

[6] Jang, H., Lim, J., Lim, J.-H., Park, S.-J., Lee, K.-C., and Park, S.-H., Finding the Evidence for Protein-Protein Interactions from PubMed Abstracts, *Bioinformatics*, 22(14):e220–e226, 2006.

[7] Kim, Y.-H., Hahn, S.-Y., and Zhang, B.-T., Text Filtering by Boosting Naive Bayes Classifiers, *Proc. 23rd Inter. ACM SIGIR Conf.*, 168–175, 2000.

[8] Schapire, R. E. and Singer, Y., Improved Boosting Algorithms Using Confidence-rated Predictions, *Machine Learning*, 37(3):297–336, 1999.