Effective Reranking for Extracting Protein-protein Interactions from Biomedical Literature

Deyu Zhou* , Yulan He , Chee Keong Kwoh

School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798

Email: zhou0063@ntu.edu.sg; asylhe@ntu.edu.sg; asckkwoh@ntu.edu.sg;

*Corresponding author

Abstract

A semantic parser based on the hidden vector state (HVS) model has been proposed for extracting protein-protein interactions. The HVS model is an extension of the basic discrete hidden Markov model, in which context is encoded as a stack-oriented state vector and state transitions are factored into a stack shift operation followed by the push of a new preterminal category label. In this paper, we investigate three different models, log-linear regression (LLR), neural networks (NNs) and support vector machines (SVMs), to rerank parses generated by the HVS model for protein-protein interactions extraction. Features used for reranking are manually defined which include the parse information, the structure information, and the complexity information. The experimental results show that reranking can indeed improve the performance of protein-protein interactions extraction, and reranking based on SVM gives more stable performance than LLR and NN.

1 Introduction

To date, vast quantity of knowledge about protein-protein interactions still hides in the full-text journals. As a result, automatically extracting these information from biomedical literature is currently one of the most interesting and difficult challenges for bioinformatics.

Much progress has been made during the past decade. Most existing approaches are either based on simple pattern matching, or by employing parsing methods. Approaches using pattern matching [1] rely on a set of predefined patterns or rules to extract protein-protein interactions. Parsing based methods employ either deep or shallow parsing. Shallow parsers [2] break sentences into none overlapping phases and extract local dependencies among phases without reconstructing the structure of an entire sentence. Systems based on deep parsing [3] deal with the structure of an entire sentence and therefore are potentially more accurate. The major drawback of the aforementioned methods is that they may require complete manual redesign of grammars or rules in order to be tuned to different domains. On the contrary, statistical models can perform the protein-protein interactions extraction task without human intervention once they are trained from annotated corpora. Many empiricist methods [4, 5] have been proposed to automatically generate the language model to mimic the features of un-structured sentences. For example, Seymore [4] used Hidden

Markov Model (HMM) for task of extracting important fields from the headers of computer science research papers. However, methods of this categories do not perform well partially due to the lack of large-scale, richly annotated corpora.

A semantic parser based on the hidden vector state (HVS) model for extracting protein-protein interactions is presented in [6]. The HVS model is an extension of the basic discrete Markov model in which context is encoded as a stack-oriented state vector. State transitions are factored into a stack shift operation similar to those of a push-down automaton followed by the push of a new preterminal category label. In this paper, we investigate three different models, log-linear regression (LLR), neural networks (NNs) and support vector machines (SVMs) to rerank parses generated by the HVS model for protein-protein interactions extraction. Experimental results show that all the three models can improve the performance of the HVS model for protein-protein interactions extraction, with reranking based on SVM giving more stable performance than LLR and NN.

The rest of the paper is organized as follows. Section 2 surveys the related work. Section 3 briefly describes the HVS model and how it can be used to extract protein-protein interactions from un-structured texts. Section 4 presents the proposed reranking methodologies. Improved experimental results are presented in section 5. Finally, section 6 concludes the paper.

2 Related work

Reranking approaches [7–10] attempts to improve upon an existing probabilistic parser by reranking the output of the parser. Reranking has benefited applications such as name-entity extraction [8], semantic parsing [10] and semantic labeling [9].

In [8], two reranking approaches, one is based Markov Random Fields, and the other is the boosting approach, were used to rerank the outputs of history-based models on Wall Street Journal corpus with a 13% relative decrease in error rate. Ge and Mooney [10] applied discriminative reranking to semantic parsing using features for reranking syntactic parses for reranking features. However, experimental results were reported that it can improve on Robocup coaching task but fail to improve on the other task-a geography database query task.

Most reranking approaches are based on discriminative models while base parers are mostly based on generative models. The reason behind is that generative probability models such as hidden Markov models, hidden vector states models provide a principled way of treating missing information and dealing with variable length sentences. On the other hand, discriminative methods such as support vector machines enable us to construct flexible decision boundaries and often result in performance superior to that of the model based other approaches. The combination of two category methods can employ merit of both methods.

3 Semantic Parser

The HVS model is a discrete Hidden Markov Model (HMM) in which each HMM state represents the state of a push-down automaton with a finite stack size. State transitions are factored into a stack shift followed by a push of one or more new preterminal semantic concepts relating to the next input word. By limiting the maximum stack depth and only allowing one new preterminal semantic concept to be pushed onto the stack for each new input word, the state space is reduced to a manageable size. The result is a model which is complex enough to capture hierarchical structure but which can be trained automatically from only lightly annotated data.

The HVS model computes a hierarchical parse tree for each word sequence W, and then extracts semantic concepts C from this tree. Each semantic concept consists of a name-value pair where the name is a dotted list of primitive semantic concept labels. In the HVS-based semantic parser, conventional grammar rules are replaced by three probability tables. Given a word sequence W, a concept vector sequence $\mathbf{C}$ and a sequence of stack pop operations N, the joint probability of $P(W, \mathbf{C}, N)$ can be decomposed as

$$P(W, \mathbf{C}, N) = \prod_{t=1}^{T} P(n_t|\mathbf{c}_{t-1}) P(c_t[1]|c_t[2 \cdots D_t]) P(w_t|\mathbf{c}_t)$$

where T is the length of the word sequence W, n_t is the vector stack shift operation, $\mathbf{c}_t$ denotes the vector state at word position t, which consists of D_t semantic concept labels (tags), i.e. $\mathbf{c}_t = [c_t[1], c_t[2], ..., c_t[D_t]]$, $c_t[1] = c_{w_t}$ is the new pre-terminal semantic label assigned to word w_t at word position t and $c_t[D_t]$ is the root concept label.

Thus, the HVS model consists of three types of probabilistic move, each move being determined by a discrete probability table:

1. popping semantic labels off the stack - $P(n|\mathbf{c})$;
2. pushing a pre-terminal semantic label onto the stack - $P(c[1]|c[2 \cdots D])$;
3. generating the next word - $P(w|\mathbf{c})$.

Each of these probability tables are estimated in training using an EM algorithm and then used to compute parse trees at run-time using Viterbi decoding. In training, each word sequence W is marked with the set of semantic concepts C that it contains. For each word w_k of each training sentence W, EM training uses the forward-backward algorithm to compute the probability of the model being in stack state c when w_k is processed. Without any constraints, the set of possible stack states would be intractably large. However, in the HVS model this problem can be avoided by pruning out all states which are inconsistent with the semantic concepts associated with W. The details of how this is done are given in [11].

4 Methodology

The procedure of employing reranking in the extraction system is described in Figure 1, where it starts with a set of sentences E. An HVS model M is constructed on E. Also a reranking model is constructed based on E, which will be described in detail later in this section. Then the sentences in the test dataset are parsed by the semantic parser M. For each sentence S_i in the test dataset, a parse set $C_i = \{C_{i1}, C_{i2}, \ldots, C_{iN}\}$ which consists of a fixed number N of candidate parses are generated. The reranking model is employed to rerank the parse set C_i for each sentence. After reranking, the top ranked parse is processed to extract protein-protein interactions.

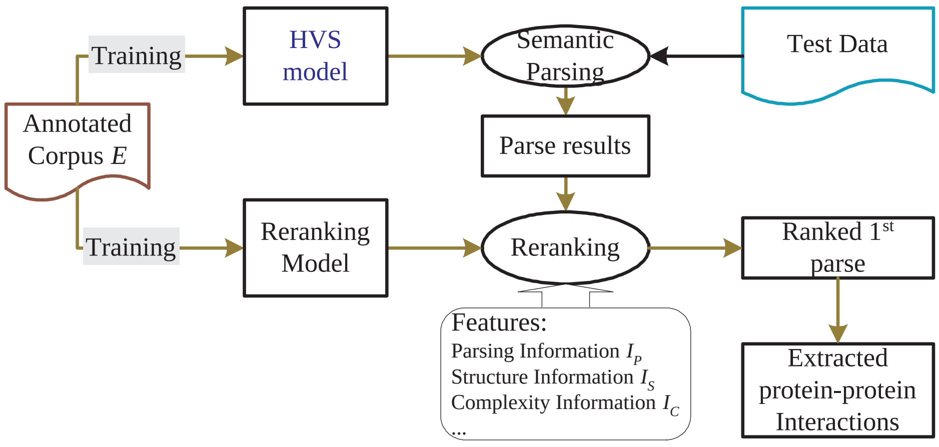

Figure 1: Semantic parse reranking for protein-protein interaction extraction.

4.1 Features for Reranking

To rerank the semantic parsing results for each sentence, we need to first select some essential features. Suppose sentence $S_i \in E$ has its corresponding parse set $C_i = \{C_{ij}, j = 1 \ldots N\}$, for each C_{ij}, its parsing information I_P, structure information I_S, complexity information I_C are defined as follows:

- **Parsing Information** I_P, describing the information in the parsing result C_{ij}, is defined as follows:

$$I_P = 1 - \frac{\sum_{k=1}^{L} KeyITD(S_{ik})}{\sum_{k=1}^{L} Key(S_{ik})}. \tag{1}$$

  Here, L denotes the length of the sentence S_i, S_{ik} denotes the kth word of the sentence S_i and the functions $KeyITD$, Key are defined as:

$$Key(S_{ik}) \quad = \quad \begin{cases} 1, \text{if } S_{ik} \text{ is a protein name or a protein interaction keyword} \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

$$KeyITD(S_{ik}) \quad = \quad \begin{cases} 1, \text{ if } Key(S_{ik}) \text{ is 1 and the semantic tag of } S_{ik} \text{ is DUMMY} \\ 0, \text{ otherwise} \end{cases} \tag{3}$$

- **Structure Information** I_S describes the similarity between the structure information of the sentence S_i and those of all the other sentences. It is defined as follows:

$$I_S = 1 - min(Dist(S_i, S_j)|S_j \in E) + \frac{Num(C(S_i))}{\|E\|}, \tag{4}$$

  where $Dist(S_i, S_j)$ describes the distance between two sentences: S_i and S_j, $C(S_i)$ denotes the cluster where S_i locates and $Num(C(S_i))$ denotes the number of sentences of E in the cluster $C(S_i)$. Here the clusters are constructed based on some standard method such as K-nearest neighborhood and the distance between two sentences is calculated based on sequence alignment. The details on how to calculate the distance between two sentences and generate the clusters can be found in [12]

- **Complexity Information** I_C, describing the complexity of the sentence S_i, is defined as follows:

$$I_C = 1 - \frac{Length(S_i)}{Max(Length(S_j)|S_j \in E)} \tag{5}$$

Overall, it can be observed that the higher the value of I_P, I_S, and I_C, the higher confidence of the correctness of the semantic parsing result C_{ij} and so the higher ranking in the parsing set C_i.

We use the above three parameters I_P, I_S and I_C to define $Score(C_{ij})$, which is the score of the parse C_{ij}. Based on the scores, parses in the parse set $C_i = \{C_{ij}, j = 1 \ldots N\}$ are reranked to a new parse set $C_i' = \{C_{io_j}, j = 1 \ldots N\}$. For all i and all $k < j$, $Score(C_{io_j}) > Score(C_{io_k})$. But the relationship between $Score$ and the three parameters I_P, I_S and I_C is unknown. It could be linear or non-linear. We therefore investigate several ways to describe this relationship which include log-linear regression, neural networks and support vector machines[1].

4.2 Log-linear Regression (LLR)

For the log-linear regression model, $Score$ is defined as

$$\log Score = \beta_p I_P + \beta_s I_S + \beta_c I_C + \beta_0, \tag{6}$$

which is a linear combination of the above defined three parameters. To estimate the coefficients $\boldsymbol{\beta} = (\beta_p, \beta_s, \beta_c, \beta_0)$, the method of least squares is applied and the coefficients $\boldsymbol{\beta}$ are selected to minimize the residual sum of squares,

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{M} (\log Score' - \beta_p I_{Pi} - \beta_s I_{Si} - \beta_c I_{Ci} - \beta_0)^2 \tag{7}$$

where M is the number of training data and $Score'$ is the true value of $Score$.

[1]We have also investigated other models such as linear regression, linear discriminant, quadratic discriminant, k-nearest neighbor and so on but no performance improvement has been observed.

4.3 Neural Network (NN)

The central idea of neural network is to extract linear combinations of the inputs as derived features, and then model the target as a nonlinear function of these features.

The model based on neural network with one hidden layer has the form:

$$\begin{aligned} Score &= \beta_0 + \sum_{m=1}^{M} \beta_m Z_m \\ &= \beta_0 + \sum_{m=1}^{M} \beta_m \sigma(\alpha_{0m} + \alpha_m^T X) \end{aligned} \tag{8}$$

where $X = (I_P, I_S, I_C)$ are inputs, Z_m are derived features created from the linear combinations of the inputs and then the target $Score$ is modeled as a function of linear combinations of the Z_m. The activation function $\sigma(v)$ is usually chosen to be the $sigmoid$ function $\sigma(v) = \frac{1}{1+e^{-v}}$. Sometimes Gaussian radial basis function are used for the $\sigma(v)$, producing a *radial basis function network*. The generic approach to fit neural networks is by gradient descent, called *back-propagation*. It can be calculated by a forward and backward sweep over the network, keeping track only of quantities local to each unit.

4.4 Support Vector Machines (SVM)

Support vector machines (SVMs) produce nonelinear boundaries by constructing a linear boundary in a large, transformed version of the feature space.

Given N pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ in training data, the SVM-based model has the form:

$$\begin{aligned} Score &= h(x)^T \beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + \beta_0 \end{aligned} \tag{9}$$

where $x = (I_P, I_S, I_C)$ are inputs. $h_m(x)$, $m = 1, \ldots, M$ are basis functions such as polynomials or splines which translate nonlinear boundaries in the original space into linear boundaries in the enlarged space to achieve better training separation. $h(x) = (h_1(x), h_2(x), \ldots, h_M(x))$ are transformed features. K, the kernel function, is the only one needs to be specified since $h(x)$ is involved only through inner products. It should be symmetric positive (semi-) definite function. Some popular choices for K are $K(x, x') = (1 + \langle x, x' \rangle)^d$, $K(x, x') = e^{-\frac{\| \| x - x' \| \|^2}{2}}$ and $K(x, x') = tanh(k_1 \langle x, x' \rangle + k_1)$.

5 Experiments

5.1 Setup

To evaluate the efficiency of the proposed method, a corpus has been constructed, which consists of sentences retrieved from the GENIA corpus [13]. GENIA is a collection of research abstracts selected from the search results of Medline database with keywords (MESH terms) *human, blood cells and transcription factors* . These abstracts were then split into sentences and those containing more than two protein names were kept. Altogether 2500 sentences were left. The corpus was split into two parts; Part I contains 1500 sentences, Part II consists of 1000 sentences which was used as the test dataset to evaluate the performance of the HVS model for protein-protein interaction.

As we mentioned before, we need to provide the abstract annotation for each sentence in the training set. An example of one sentence, its annotation and its contained protein-protein interactions which we attempt to extract is given as following:

Sentences:	CUL-1 was found to interact with SKR-1, SKR-2, SKR-3, SKR-7, SKR-8 and SKR-10 in yeast two-hybrid system.
Annotation:	PORTEIN_NAME(ACTIVATE (PROTEIN_NAME))
PPI :	CUL-1#SKR-1#ACTIVATE
	CUL-1#SKR-2#ACTIVATE
	CUL-1#SKR-3#ACTIVATE
	CUL-1#SKR-7#ACTIVATE
	CUL-1#SKR-8#ACTIVATE
	CUL-1#SKR-10#ACTIVATE

We have manually made annotation for sentences in Part I.

To train the reranking models, firstly we randomly selected 500 sentences from Part I, then used the base parser trained on Part I to output 50 parses for each sentence.

Since these randomly selected 500 sentences from Part I have been manually annotated, for each sentence, the score (correctness) of the 50 parses can be easily evaluated by comparing these parses with its annotation and its protein-protein interactions. For example, the program which has been used to extract protein-protein interaction from these parses can be adopted to evaluate the score of these parses. The score of one parse is correlated with F-measure of extracting performance from the parse. If all these protein-protein interactions in the sentence can be extracted without generating wrong protein-protein interactions from the parse, the score of the parse is defined as 1, the same as the F-measure of the parse. Two parse were selected from the 50 parses to form the training instances, one with the highest score and the other with the lowest score. Altogether 1000 training instances were chosen to form the training dataset for the reranking models.

Three reranking models were investigated. The log-linear regression (LLR) model was implemented using the S-plus software. The neural network (NN) model was implemented using Matlab which consists of two layers, layer one having 10 neurons and layer two having 1 neuron. Both LLR and NN were trained on 1000 training instances. The support vector machine (SVM) model was implemented using SVM^{light} [14]. It was trained in a different way. For the selected 500 sentences from Part I, 500 SVM models were built with each model being trained on the best and the worst parses from each corresponding sentence. The models were then used to predict scores for the parses generated from the test dataset. The final score of a parse is the average of all the scores predicted by the 500 models. The parse with the highest score is then selected as the output.

The results reported here are based on the values of TP (true positive), FN (false negative), and FP (false positive). TP is the number of correctly extracted interactions. (TP+FN) is the number of all interactions in the test set and (TP+FP) is the number of all extracted interactions. F-measure is computed using the formula below:

$$\text{F-measure} = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \tag{10}$$

where Recall is defined as $TP/(TP+FN)$ and Precision is defined as $TP/(TP+FP)$.

5.2 Results

The semantic parser M trained on Part I was used to output 50 parses for each sentence in the test dataset. A set of experiments were first conducted to discover the optimal number of candidate parses that could be used for reranking. Figure 2 shows the protein-protein interactions extraction performance versus the number of candidate parses. Interestingly, the best performance was obtained when the number of candidate parses is set to 8 where all the three reranking models give the F-measure value around 59.8%. Increasing the number of candidate parses did not give the improved performance. The results generated by SVM are fairly stable with F-measure flattening around 59.4%. The performance obtained from LLR and NN fluctuates within the range between 58.6% and 59.4%.

Table 1 lists the best extraction performance for each method. The "baseline" results were obtained using the initial HVS model trained on E (1500 sentences) without employing reranking methods. The "SVM" results were obtained using the HVS model employing the SVM reranking method. The "NN" row shows the performance of the HVS model employing the Neural Network reranking method. The "LLR" row shows the performance of the HVS

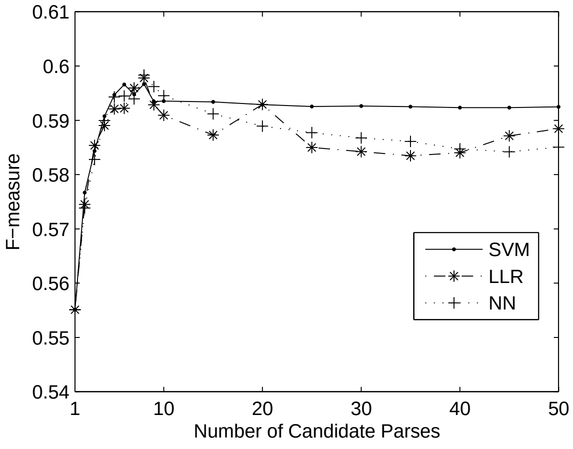

Figure 2: F-measure vs number of candidate parses.

model employing the log-linear regression reranking method. It can be observed that employing reranking did give the performance gain of 7% relatively compared to the baseline model.

Table 1: Protein-protein interaction extraction results from the HVS model with or without employing different reranking methods.

Experiment	Recall (%)	Precision (%)	F-Score (%)
Baseline	55.8	55.6	55.7
SVM	59.1	60.2	59.7
NN	57.9	61.8	59.8
LLR	58.5	61.2	59.8

5.3 Discussion

The reranking models we implemented here can be categorized into two types, the generative models and the discriminative models, with LLR and NN being generative models and SVM being a discriminative one. Since the HVS model is a generative model as mentioned in section 1, in general, parsing performance should be improved by discriminative reranking models instead of generative reranking models following received wisdom. However results in our experiment show that both the generative and discriminative reranking approaches can improve the protein-protein interactions extraction performance with the discriminative reranking approach such as SVM giving more stable performance.

The relative improvement 7% in F-measure is however only marginal. The possible reasons behind are analyzed as follows. Firstly, the three manually defined features may not fully describe the relationship between the reranking order and the parse. In future work, we would consider employing both syntactic and semantic information to define the features. Secondly, the limited size of the training data for the HVS model makes the reranking methods less effective. Increasing the amount of the training data should further improve the reranking performance.

6 Conclusion

This paper has presented three reranking methods for the HVS model in the application of extracting protein-protein interactions from biomedical literature. Experimental results show that 4% relative improvement in F-measure can be obtained through reranking on the semantic parse results. This is very encouraging. Future research could be conducted in several aspects. For example, currently used manually defined features might not be sufficient to discriminate the best parse and the rest candidate parses. Incorporating other semantic or syntactic information might be able to give further gains. Besides reranking the parse results, discriminatively training the HVS model which is originally trained based on maximum likelihood estimation criteria might also improve the performance.

References

1. Huang M, Zhu X, Hao Y: **Discovering patterns to extract protein-protein interactions from full text**. *Bioinformatics* 2004, **20**(18):3604–3612.

2. Pustejovsky J, Castano J, Zhang J, Kotecki M, Cochran B: **Robust Relational Parsing Over Biomedical Literature: Extracting Inhibit Relations**. In *Proceedings of the Pacific Symposium on Biocomputing.*, Hawaii, U.S.A 2002:362–373.

3. Temkin JM, Gilder MR: **Extraction of protein interaction information from unstructured text using a context-free grammar**. *Bioinformatics* 2003, **19**(16):2046–2053.

4. Seymore K, McCallum A, Rosenfeld R: **Learning Hidden Markov Model Structure for Information Extraction**. In *AAAI 99 Workshop on Machine Learning for Information Extraction* 1999.

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

6. Zhou D, He Y, Kwoh CK: **Extracting Protein-Protein Interactions from the Literature using the Hidden Vector State Model**. In *International Workshop on Bioinformatics Research and Applications, LNCS 3992*, Reading, UK 2006:718–725.

7. Collins M: **Discriminative Reranking for Natural Language Parsing**. In *Proceeding of 17th International Conference on Machine Learning (ICML)*, Morgan Kaufmann, San Francisco, CA 2000:175–182.

8. Collins M: **Ranking algorithms for named-entity extraction: Boosting and the voted perceptron**. In *Proceedings of the Annual meeting of the Association for Computational Linguistics (ACL) 2002* 2002:489–496.

9. Toutanova K, Haghighi A, Manning CD: **Joint Learning Improves Semantic Role Labeling**. In *Proceedings of the Annual meeting of the Association for Computational Linguistics (ACL) 2005* 2005:589 – 596.

10. Ge R, Mooney RJ: **Discriminative Reranking for semantic Parsing**. In *Proceedings of the conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL) 2006* 2006:263–270.

11. He Y, Young S: **Semantic Processing using the Hidden Vector State Model**. *Computer Speech and Language* 2005, **19**:85–106.

12. Zhou D, He Y, Kwoh CK: **Training the Hidden Vector State Model from Un-annotated Corpus**. In *International Conference on Computational Science, Advanced Computational Approaches and IT Techniques in Bioinformatics*, Beijing, China 2007.

13. Kim J, Ohta T, Tateisi Y, Tsujii J: **GENIA corpus–semantically annotated corpus for bio-textmining**. *Bioinformatics* 2003, **19**(Suppl 1):i180–2.

14. Joachims T: *Learning to Classify Text Using Support Vector Machines* 2002.