Uncovering Protein-Protein Interactions in the Bibliome

Alaa Abi-Haidar[1,6], Jasleen Kaur[1], Ana Maguitman[2], Predrag Radivojac[1], Andreas Retchsteiner[3], Karin Verspoor[4], Zhiping Wang[5], Luis M. Rocha[1,6] *

*To whom correspondence should be addressed: rocha@indiana.edu

[1] School of Informatics, Indiana University, USA
[2] Dep. de Ciencias e Ing. de la Computación, Universidad Nacional del Sur, Argentina
[3] Center for Genomics and Bioinformatics, Indiana University, USA
[4] Information Sciences Group, Los Alamos National Laboratory, USA
[5] Biostatistics, School of Medicine, Indiana University, USA
[6] FLAD Computational Biology Collaboratorium, Instituto Gulbenkian de Ciência, Portugal

Abstract

We participated in three of the Protein-Protein Interaction (PPI)subtasks: Protein Interaction Article Sub-task 1 (IAS), Protein Interaction Pairs Sub-task 2 (IPS), and Protein Interaction Sentences Sub-task 3 (ISS). Our approach includes a feature detection method based on a spam-detection algorithm. For IAS we submitted three runs from distinct classification methods: the novel Variable Threshold Protein Mention Model, Support Vector Machines, and an integration method based on measures of uncertainty and a nearest neighbor predictor on the eigenvector space obtained via the Singular Value Decomposition of the feature/abstract matrix. For IPS and ISS we used the features discovered from IAS abstracts as well as features from training IPS and ISS data to predict appropriate passages and pairs. We also used the number of protein mentions in a passage as a feature.

Keywords: Protein interaction, text mining, bibliome informatics, support vector machines, singular value decomposition, spam detection, uncertainty measures, proximity graphs, complex networks.

1 Protein Interaction Article Sub-Task 1 (IAS)

1.1 Feature Selection

All three runs submitted use word features extracted from the training data using a method inspired by the spam filtering system *SpamHunting* (Fdez-Riverola et al., 2007). First, we computed the probability that a word w appears on a positive $p_{TP}(w)$ abstract, as the ratio of the number of positive abstracts containing w, over the total number of positive abstracts. Similarly, we computed the probability that a word w appears on a negative abstract $p_{TN}(w)$. After stemming with the Porter algorithm, filtering out short words with 2 or less letters, and removing common stop words except the word "with", we ranked all words according to the score: $S(w) = |p_{TP}(w) - p_{TN}(w)|$. The words with the highest score S tend to be associated either with positive or negative abstracts. Therefore, such words are assumed to be good features for classification.

1.1.1 Single Word Feature Set

The first feature set we used were the top 650 stemmed abstract words with largest S; the top 15 words are listed in table 1 in the supplemental materials (section 3 and online [1]), which also includes figure 3 depicting the 1000 abstract words with largest S in the space of p_{TP} and p_{TN}.

[1] http://informatics.indiana.edu/rocha/bc2

1.1.2 Word Pair Feature Sets

We produced two additional feature sets comprised of word pairs obtained from the 650 stemmed word features in the first set. This leads to $650^2 = 422500$ possible word pairs, though not all occur. First, we removed all words not in the first feature set from the abstracts. Then, from the filtered abstracts we obtain the second and third feature sets, which are comprised of pairs of words immediately adjacent (bigrams) and pairs of words that occur within a window of ten words from each other, respectively. We also computed the probability that such word pairs (w_i, w_j) appear in a positive or negative abstract: $p_{TP}(w_i, w_j)$ and $p_{TN}(w_i, w_j)$, respectively. Figure 4 depicts the 1800 word pairs of the third feature set with largest: $S^{10}(w_i, w_j) = |p_{TP}(w_i, w_j) - p_{TN}(w_i, w_j)|$. Table 1 in the supplemental materials (section 3) lists the top 15 word pairs for S^{10}.

1.1.3 Number of Protein Mentions

Using *A Biomedical Named Entity Recognizer* (ABNER) [2] (Settles, 2005), we extracted unique protein mentions from abstracts. These mentions were later converted to UniProt IDs only for the IPS and ISS tasks (see section 2); for the IAS task we used the number of unique ABNER protein mentions per abstract a, $np(a)$, as an additional feature or parameter.

1.2 Classification Methods

To test the various classification methods described below, we performed k-fold tests on the supplied training data, as well as additional data from MIPS (positives) and abstracts curated by hand (negatives) that were graciously donated to our team by Santiago Schnell. Based on the results of these tests, we submitted the three runs described below.

1.2.1 Support Vector Machine Classification

Starting from the first feature set (single words with largest S) we applied additional dimensionality reduction and then trained classification models to discriminate between positive and negative data. Dimensionality reduction involved a two-step process. First, a feature selection filter based on the t-test was used in which all features with the p-value below a pre-specified threshold t_f were retained. Then, we applied the principal component analysis (Wall et al., 2003) to retain all features containing $t_{PCA} \cdot \sigma^2$ of the total variance σ^2. The remaining features were fed into a support vector machine, a classification model used to maximize the margin of separation between positive and negative examples (Vapnik, 1998). We used the $\texttt{SVM}^{light}$ package (Joachims, 2002) in which we explored both polynomial and Gaussian kernels with various parameters. The overall system was trained to maximize the classification accuracy on the unlabeled data using the following two-step iterative procedure: (i) train a classifier with costs adjusted to the current estimates of class priors in the unlabeled data; and (ii) predict class labels on the unlabeled set using current classifier and make new estimates of the class priors. Initially, class priors in the unlabeled data were set to 0.5. Not more than five rounds were executed, ending with the total cost of positive examples being about 3 times the costs of the negatives. The final predictor used $t_f = 0.1$ for the feature filtering, $t_{PCA} = 0.95$ for the principal component analysis and a linear support vector machine.

1.2.2 Variable Trigonometric Threshold Classification

We developed trigonometric measures of term relevance on the p_{TP}/p_{TN} plane. It is obvious that the best features in this plane are the ones that are closest to either one of the axis. Any feature w is a vector in this plane (see figure 1); let α be the angle of this vector with the p_{TP} axis. We then

[2] http://www.cs.wisc.edu/~bsettles/abner/

use $\cos(\alpha)$ as a measure of feature terms [3] mostly associated with positive abstracts, and $sin(\alpha)$ of terms mostly associated with negative ones (in the training data set). Then, for every abstract a, we compute the sum of all feature term contributions for a positive (P) and negative (N) decision:

$$P(a) = \sum_{w \in a} \cos(\alpha(w)), \quad N(a) = \sum_{w \in a} \sin(\alpha(w)) \tag{1}$$

The decision of whether abstract a is a positive or negative abstract (in so far as being relevant for protein-protein interaction) is then computed as:

$$\begin{cases} a \in TP, & \text{if } \frac{P(a)}{N(a)} \geq \lambda_0 + \frac{\beta - np(a)}{\beta} \\ a \in TN, & \text{otherwise} \end{cases} \tag{2}$$

where λ_0 is a constant threshold for deciding whether an abstract is positive (relevant) or not. This threshold is subsequently adjusted for each abstract a with the factor $(\beta - np(a))/\beta$, where β is another constant, and $np(a)$ is the number of protein mentions in abstract a as described in section 1.1.3. We observed that abstracts have a higher chance of being positive (relevant) with more protein mentions, thus, via formula 2, the classification threshold is linearly decreased as np increases. This means that with a high (lower) number of protein mentions, it is easier to classify an abstract as positive (negative). When $np(a) = \beta$ the threshold is simply λ_0. We refer to this classification method as *Variable Trigonometric Threshold* (VTT).

The specific value of λ_0 was determined by optimizing the F-Score measure[4] on the training data as well as on the additional abstracts obtained from MIPS and hand curated. To decide on the best β we computed the probability that a positive abstract a in the training set contains more than np protein mentions: $pos = p(TP|np)$. We also computed the negative counterpart: $neg = p(TN|np)$. We observed that when $np \geq 7$, we maximize $pos - neg$, thus we set $\beta = 7$. This way, when $np(a) > 7$ the decision threshold is lowered, and raised otherwise. Figures 5 and 6 in the supplemental materials (section 3) depict this study. Finally, the run we submitted with VTT used the following parameters: $\lambda_0 = 1.7$ and $\beta = 7$, using the top 650 word-pair features of the third feature set (section 1.1.2). This combination of parameters resulted in the best F-Score values for the training and additional data.

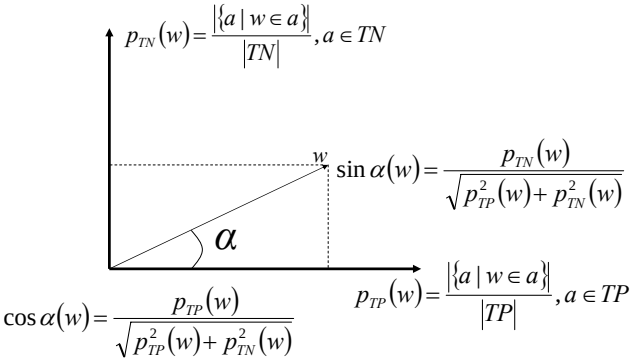

Figure 1: Trigonometric measures of term relevance for identifying positive and negative abstracts in the P_{TP} and P_{TN} plane.

1.2.3 Classification with Singular Value Decomposition Plus Uncertainty Integration

We first classified the set of abstracts using a nearest neighbor classifier on the eigenvector space obtained via the *Singular Value Decomposition* (SVD) (Wall et al., 2003) of the feature/abstract space. We used the first feature set of single word features (section 1.1.1). We represented abstracts as vectors in this feature space. We then calculated the inverse document frequency measure (IDF), so the vector coefficients were the TF*IDF (Dumais, 1990) for the respective features. The number of protein mentions per abstract a, $np(a)$ (section 1.1.3), was added as an additional feature. The abstract vectors were also normalized to Euclidean length 1. We computed the SVD of the resulting abstract-feature

[3] By term, we refer to features in our three different feature sets as described in section 1.1.

[4] F-measure is defined as F = $\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$, where $Precision$ is the proportion of abstracts returned that are relevant (positive), and $Recall$ is the proportion of relevant abstracts that are retrieved.

matrix (from the training data). The top 100 components were retained (this number provided best results on our tests on training and additional data). To classify a test abstract vector a, we project it onto this SVD subspace and calculate the cosine similarity measure of a to every training abstract t:

$$\cos(a,t) = \frac{a.t}{||a|| \times ||t||} \tag{3}$$

We then calculate positive and negative scores for each test abstract a by summing the cosine measure for every positive ($t \in TP$) and negative ($t \in TN$) training abstract, respectively:

$$P(a) = \frac{1}{|TP|}\sum_{t\in TP}\cos(a.t), \quad N(a) = \frac{1}{|TN|}\sum_{t\in TN}\cos(a.t) \tag{4}$$

where $|TP|$ and $|TN|$ are the number of positive and negative abstracts in the training data, respectively. Finally, a linear decision boundary was determined in the two-dimensional space of P and N: abstract a is classified as positive (relevant) if $P(a) > m \cdot N(a) + b$ and as negative otherwise. Coefficients m and b were determined manually from optimizing the F-Score measure on the training data as well as on the additional abstracts obtained from MIPS and hand curated. Figure 7 in the supplemental materials (section 3) depicts the boundary surface in the P and N space.

Using a variation of a method we previously used (Maguitman et al., 2006), we integrated three variations of the VTT classification (section 1.2.2) with the SVD classification in such a way that for each abstract the most "reliable" prediction was used to issue a classification. To ascertain reliability, we represented the target test abstract a, as well as all abstracts t in the training data, as vectors in a compound feature space (including all three feature sets described in section 1.1). Next, we computed the cosine similarity, $cos(a,t)$, between a target a and every t, and treated this value as a weighted vote. Thus, if abstract t is very close to a, it will have a greater influence in the classification of a. Since for any abstract t in the training data, we know if a classification method correctly classified it, we tried two different measures or reliability:

- **Entropy-Based Measure:** As in (Maguitman et al., 2006), we used Shannon's entropy to compute the uncertainty of a prediction for the target abstract based on the distribution of positive and negative weighted votes obtained for that abstract from a given classification method.

  Let $\rho_M(a,TP)$ and $\rho_M(a,TN)$ be the probabilities of predicting TP or TN, respectively, as the class for abstract a using method M. We estimate these probabilities as follows:

  $$\rho_M(a,TP) = \frac{\sum_{t\in TP} cos(a,t)}{\sum_{t\in TP\cup TN} cos(a,t)}, \quad \rho_M(a,TN) = \frac{\sum_{t\in TN} cos(a,t)}{\sum_{t\in TP\cup TN} cos(a,t)}.$$

  Note that $\rho_M(a,TP) = 1 - \rho_M(a,TN)$. Finally, we compute the prediction uncertainty of abstract a using method M, $U_M(a)$, using Shannon's entropy as follows:

  $$U_M(a) = -\rho_M(a,TP)\log\rho_M(a,TP) - \rho_M(a,TN)\log\rho_M(a,TN)$$

  Using the uncertainty measure we integrate the predictions issued by each method by selecting, for each abstract a, the prediction issued by the method M with lowest $U_M(a)$

- **Misprediction Measure:** We used the information about correct predictions available from the training set to compute a *misprediction* rate from each classification method; each neighbor t of the target abstract a contributed to a method's rate based on its weighted vote.

  Assume T is the training set of abstracts, and $I^M \subseteq T$ be the set of abstracts that has been misclassified using method M. Let $\mu_M(a)$ be the misprediction rate for abstract a based on the weighted votes for a from abstracts $t \in I^M$:

  $$\mu_M(a) = \frac{\sum_{t\in I^M} cos(a,t)}{\sum_{t\in T} cos(a,t)}$$

Using this misprediction rate we integrate the predictions issued by each method by selecting, for each abstract a, the prediction issued by the method M with lowest $\mu_M(a)$

Finally, we calculated the product of the Entropy-Based Measure and the misprediction Measure and selected, for each target test abstract a, the prediction coming from the classification method with lowest product. In our submission for run 3, we used this uncertainty-driven integration with the following four classification methods:

1. SVD Vector model with first feature set of single words.
2. Fixed Threshold Classification (FT). This is the same as the VTT classification (section 1.2.2) but without trigonometric measures. In this case, instead of the formulae 1, we simply used:

$$P(a) = \sum_{w \in a} p_{TP}(w), \quad N(a) = \sum_{w \in a} p_{TN}(w) \tag{5}$$

   We also did not use the ABNER protein mention counts, thus formula 2 becomes simply $P(a)/N(a) > \lambda_0 = 1.3$. In this case, we also used the first feature set of single words.
3. VTT exactly as described in section 1.2.2, but with the second feature set (bigrams) and $\lambda_0 = 1.5$ and $\beta = 7$.
4. VTT exactly as described in section 1.2.2.

Items 2 to 4 were chosen so that there would be a model for each of the three feature sets. The specific parameters were chosen from the F-score performance with the learning and additional data. It is important to notice that in our tests, the uncertainty-driven integration algorithm (SVD-UI) improved only very slightly over the SVD vector model alone. Indeed, for the test set the SVD vector model alone produced the same classification as the integration method, except that different rankings of abstracts were attained. We decided to submit the results of the integration method because it slightly improved on the SVD vector model with the learning data.

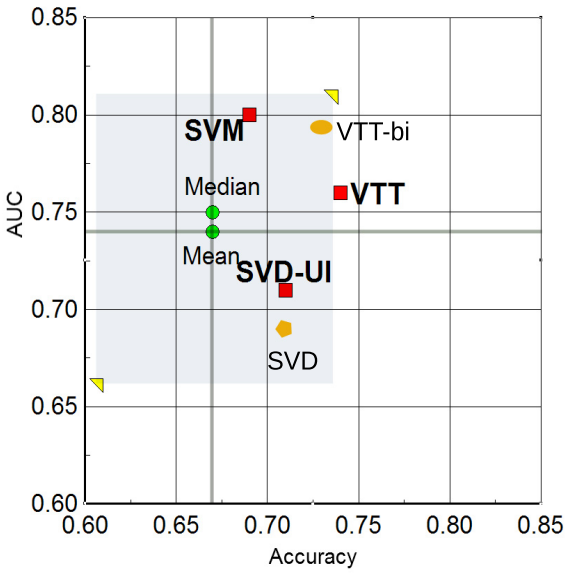

Figure 2: Our methods on the accuracy vs. AUC plane for IAS. Mean and Median are for the set of all submissions from all groups. Red squares denote our three submissions (SVM, VTT, and SVD-UI). The orange polygon denotes the results for SVD alone, and the orange oval denotes the results for one of the versions of VTT (with bigrams) included in the SVD-UI method.

1.3 Results

The performance of the three runs we submitted (sections 1.2.1, 1.2.2, and 1.2.3) can be seen in Table 2 of the supplemental materials (section 3). The three runs produced similar results regarding the F1 measure (*F-Score*), with the highest value (0.75) for Run 2 (VTT, section 1.2.2), and lowest (0.73) for Run 3 (SVD-UI, section 1.2.3). However, this measure hides the distinct capabilities of each method. Indeed, the SVM method resulted in the best recall and worst precision (0.88/0.64), whereas the VTT method resulted in the worst recall and best precision (0.79/0.71). The SVD-UI method lies in between the other two (0.8/0.68), though its F-Score measure is slightly worse.

Perhaps a better measure for this task is *accuracy*, which gives us the ratio of correct predictions (for both positive and negative abstracts). In this case, the VTT method yielded the best result (0.74),

followed by the SVD-UI (0.71), and the SVM (0.69) methods. Thus, the VTT method lead to a more balanced prediction for both positive and negative abstracts, leading to the lowest error rate (0.26).

When we look at the *Area Under the ROC Curve* (AUC) measure, however, the results have yet another reading. This measure can be understood as the probability that for a randomly picked positive abstract and a randomly picked negative abstract, the positive abstract is ranked above the negative one. We obtained very good results with this measure for the SVM run (0.8), followed by good results for the VTT (0.76) and SVD-UI (0.71) methods. Contrasting accuracy with AUC, we observe that while the VTT method lead to our highest accuracy, the probability of finding a false positive closer to the top of the rank (or a false negative closer to the bottom of the rank) is larger than in the ranking produced by the SVM method (see figure 8). This situation was even worse with the SVD-UI method, as can be seen in figure 9, where many negative (positive) abstracts appear deep in the positive (negative) side of the decision surface. Conversely, while the SVM method lead to our lowest accuracy measure, it yielded the highest AUC, which indicates that a larger proportion of its erroneous decisions were closer to its decision surface.

As we discuss in section 1.2.3, the SVD vector model alone produced the same classification as SVD-UI, except that different rankings of abstracts were attained. We note that the AUC of the SVD method alone was lower3(0.68) than that of the AUC-UI method (0.71). We can thus say that the integration method improved the AUC of the SVD method alone. However, it produced worse AUC and accuracy values for other constituent methods, such as VTT as submitted in Run 2. Indeed, the AUC and accuracy of the not submitted individual methods included in the uncertainty integration method (section 1.2.3), show that constituent method 3, a version of VTT method using bigrams, produces a higher AUC (0.79) than the VTT we submitted and the SVD-UI method, without sacrificing accuracy much (0.73). This means that the integration method did worse than some of its constituents, and that the VTT method can produce better results. A comparison of all our methods in the AUC/Accuracy plane is depicted in Figure 2. The figure also contrasts our results with the central tendency of all group submissions. The most salient points are:

- **Accuracy**: All three runs are above the mean and median values of accuracy for all teams. Run 2 (VTT) yielded an accuracy above one standard deviation of the mean accuracy.
- **AUC**: Both the SVM and VTT methods are above the mean and median value of AUC, but the SVM method is very nearly above one standard deviation above the mean.
- **Balance across all performance measures**: The VTT method was the only one which was above the mean for all measures tested (precision, recall, F-score, accuracy, and AUC).

2 Protein Interaction Pairs And Sentences Sub-Tasks (IPS AND ISS)

2.1 Feature Selection

From the IAS subtask, we collected the top 1000 word-pair features, (w_i, w_j) from the third feature set (section 1.1.2). Since the purpose of these tasks is to identify portions of text where protein-protein-interaction (PPI) information appears, we do not need to worry about features indicative of negative PPI information. Features are chosen and ranked according to high value of:

$$p(w_i, w_j) = p_{TP}(w_i, w_j).cos(\alpha(w_i, w_j)) = \frac{p_{TP}^2(w_i, w_j)}{\sqrt{p_{TP}^2(w_i, w_j) + p_{TN}^2(w_i, w_j)}} \tag{6}$$

where p_{TP} and p_{TN} are as defined in section 1.1. We multiply the cosine measure by the probability of the feature being associated with a positive abstract, to ensure that features which have zero probability of being associated with a negative abstract (there are many), are not equally ranked with $p(w_i, w_j) = 1$. We refer to this set of 1000 stemmed word pairs, as the ***word pair feature set***.

We also obtained an additional set of features from PPI-relevant sentences: the ***sentence feature set***. These sentences were extracted from the various files provided by Biocreative for these tasks We collected all sentences that contained PPI, and calculated the frequency of stemmed words in this collection: $f_{ppi}(w)$. Then we calculated the frequency of stemmed words of the entire corpus: $f_c(w)$. Finally, similarly to the word pair features above, we selected as sentence features the top 200 words which maximize the following score (top 10 listed in Table 3.):

$$SS = \frac{f_{ppi}^2(w)}{\sqrt{f_{ppi}^2(w) + f_c^2(w)}} \tag{7}$$

2.2 Paragraph Selection and Ranking

We used our two feature sets plus protein mention information to select the paragraphs in each document which are more likely to contain PPI information. Thus, for each full text document, we ranked paragraphs according to three different criteria:

A Largest sum of word pair feature weights (section 2.1), where the weights are the inverse feature rank. Paragraphs without feature matches are thrown out (rank 0).

B Largest number of protein mentions in paragraph. As in the IAS subtask (see section 1.1.3), we also used ABNER to collect protein mentions in the full text documents provided for these two subtasks. Paragraphs without protein mentions are thrown out (rank 0).

C Largest number of sentence features in paragraph (section 2.1). Each feature that occurs in a paragraph adds 1 to the count. Paragraphs without feature matches are thrown out (rank 0).

From these three distinct paragraph rankings, for each document, we produced another three rankings that aim to integrate this information in different ways. For each document, we rank paragraphs according to the following criteria:

1. Rank product of ranks produced in A (word pair features) and B (protein mentions) above.
2. Rank product of ranks produced in B (protein mentions) and C (sentence features) above.
3. Rank product of ranks produced in A, B, and C above.

Since paragraphs thrown out in A, B and C are rank 0, in this step, only paragraphs with feature matches and protein mentions remain. The resulting 3 rankings constitute the paragraph rankings in the three runs submitted for the IPS subtask: 1, 2, and 3, respectively.

2.3 Mapping of Protein Mentions to UniProt IDs

To obtain the actual protein-protein interaction pairs contained in the paragraphs of ranks 1,2, and 3 described in section 2.3, we had to convert the textual mentions obtained with ABNER to UniProt IDs. Protein and gene references identified using the ABNER system were mapped to UniProt IDs through exact matching with either a gene or a protein name occurring in SwissProt—considering both primary names and synonyms. UniProt version 8.2 was used for the mapping; this is not the most current version and could have resulted in missing relevant mappings. These mappings were then filtered using the provided UniProt subset. This process typically resulted in many UniProt IDs for the same ABNER protein mention, mostly because the same protein name maps to different UniProt IDs for different organisms. We therefore filtered the protein mention to include only UniProt ID mappings associated with organisms in the set of MeSH terms of a given article. Unfortunately, many of the articles listed several organisms in their MeSH keyterms. An obvious improvement would be to detect the appropriate organism for a given paragraph more specifically.

2.4 Selection and Ranking of Protein-Protein Interaction Pairs for IPS

Finally, for the IPS task, we returned all the combinations of protein pairs (UniProt accession numbers as described in section 2.3) occurring in the same sentence—for sentences included in the paragraphs of ranks 1,2, and 3 (section 2.2). For a given document (PMID), the rank of each protein-protein interaction pair is the rank of the highest ranked paragraph in which the pair occurs in a sentence. We submitted three distinct rankings of PPI pairs according to the three ranks 1,2, and 3 (section 2.2).

2.5 Protein Mention Feature Expansion with Proximity Networks

We used a method we employed in the first Biocreative competition to obtain additional, contextualized features associated with a protein names (Verspoor et al., 2005), that is, additional features which are relevant in a specific document, but not necessarily in the whole corpus. We computed for each document a word proximity network based on a co-occurrence proximity measure of stemmed words in paragraphs of that document:

$$WPP(w_i, w_j) = \frac{\sum_{k=1}^{m} (r_{i,j} \wedge r_{i,j})}{\sum_{k=1}^{m} (r_{i,j} \vee r_{i,j})} \tag{8}$$

where $r_{i,j} \in \{0, 1\}$ is an element of the relation $R : P \times W$; P is the set of all m paragraphs in a document, and W is the set of all n stemmed words. This yields a proximity network for each document, where the nodes are words w_i, and the edges are the $WPP(w_i, w_j)$ proximity weights.

Next, for every PPI pair (obtained by rank 1) occurring in a given document, we obtain the words closest to the protein labels in the network. Notice that these protein labels are words identified by ABNER for the given PPI pair, and they appear on the proximity network as regular nodes. For each protein pair we selected the 5 stemmed words (nodes) in the proximity network with largest minimum proximity to both protein names. The sentences in the articles where the PPI pairs occur were then augmented using the 5 words obtained from the relevant document.

2.6 Passage Extraction and ISS Submission

From ranked paragraphs, we selected passages (sets of 3 sentences) containing a given PPI pair. Finally, we submitted three runs to the ISS subtask:

1. Passages ranked by largest number of occurring word pair features (see section 2.1).
2. Passages ranked by largest number of occurring word pair features, but where the PPI occurring sentence is expanded with words from the document's proximity network.
3. Same as 2, with the addition of the following factor 100/paragraph_rank_1 (see section 2.2) to the number of features found in the passage.

2.7 Results

The results for the IPS and ISS tasks were disappointing, though in line with the central tendency of all submissions. Our three submitted runs to IPS were hardly distinguishable. For all our three runs, the precision was below the mean and median of all submissions, but still well within one standard deviation of the mean. On the other hand, recall was above the mean and median of all submissions, and above one standard deviation of the mean. The F-score was very near the mean and median of all submissions. These results were true for both the identification of protein-protein interaction pairs (PPIN) and single interactors (PN), as well as for the set of all articles (All) and the subset of articles

containing exclusively SwissProt interaction pairs (SP). Figure **??** in the supplemental materials lists the specific values.

Regarding the ISS subtask, the three submitted runs were slightly different, and denoted a slight improvement with the number of the run. Run 2 was better than run 1, which shows that the proximity expansion improved a little the original features. Run 3 was better than run 2, showing that considering the paragraph rank from IPS (which includes number of protein mentions) in addition to the expanded word-pair features is advantageous. Again our results were in line with the averaged values of all submissions. Our matches (387) and unique matches (156) to previously selected passages were above the average of all submissions (207.46 and 128.62, respectively). We should notice, however, that we predicted many more passages (18371) and unique passages (5252) than the average (6213.54 and 3429.65, respectively), which lead to lower than average fractions of correct from predicted and unique passages. Like in the IPS case, this means that our system was better at recall than at precision. Finally, our mean reciprocal rank of correct passages substantially higher than average (0.66 to 0.56). Table 4 in the supplemental materials lists the specific values.

Acknowledgements

We would like to thank Santiago Schnell for graciously providing us with additional proteomics related articles not containing protein-protein interaction information. We would also like to thank the FLAD Computational Biology Collaboratorium at the Gulbenkian Institute in Oeiras, Portugal, for hosting and providing facilities used to conduct part of this research during the summer of 2006. We are grateful to Indiana University's Research and Technical Services for technical support. The AVIDD Linux Clusters used in our analysis are funded in part by NSF Grant CDA-9601632.

References

Dumais, S. (1990). Enhancing performance in latent semantic indexing. Technical Report TM-ARH-017527, Bellcore.

Fdez-Riverola, F., Iglesias, E., Diaz, F., Mendez, J., and Corchado, J. (2007). Spamhunting: An instance-based reasoning system for spam labelling and filtering. *Decision Support Systems*, In Press.

Joachims, T. (2002). *Learning to classify text using support vector machines: methods, theory, and algorithms*. Kluwer Academic Publishers.

Maguitman, A., Rechtsteiner, A., Verspoor, K., Strauss, C., and Rocha, L. (2006). Large-scale testing of bibliome informatics using pfam protein families. In *Pacific Symposium on Biocomputing, Vol. 11*, pages 76–87.

Settles, B. (2005). Abner: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21(14):3191–3192.

Vapnik, V. (1998). *Statistical learning theory*. John Wiley & Sons, New York.

Verspoor, K., Cohn, J., Joslyn, C., Mniszewski, S., Rechtsteiner, A., Rocha, L., and Simas, T. (2005). Protein annotation as term categorization in the gene ontology using word proximity networks. *BMC Bioinformatics*, 6 Suppl 1:S20.

Wall, M., Rechtsteiner, A., and Rocha, L. (2003). Singular value decomposition and principal component analysis. In Berrar, D., Dubitzky, W., and Granzow, M., editors, *A Practical Approach to Microarray Data Analysis*, pages 91–109. Kluwer, Norwell,MA.