Automatically Annotating Documents with Normalized Gene Lists

Jeremiah Crim, Ryan McDonald and Fernando Pereira
Department of Computer and Information Science
University of Pennsylvania
Levine Hall, 3330 Walnut Street, Philadelphia, PA 19104
{jcrim,ryantm,pereira}@cis.upenn.edu

1 Introduction

Recently, researchers have begun to apply machine-learning-based information extraction techniques to biomedical text, most notably to identify gene and chemical compound mentions [5, 7, 8]. A related problem is gene normalization, which involves annotating a document (or abstract) with the list of genes that are mentioned within it. Gene normalization requires as input a *synonym list*. Each entry in the list represents a specific gene and contains both a unique identifier for that gene (known as a normal form) and a set of different ways in which the gene may be mentioned. Ideally this list will be complete. However, in practice, gene mentions vary widely and evolve over time. Table 1 contains a few entries of the synonym list for the fly organism.

To facilitate gene normalization research, the organizers of BioCreative [1] have made available 5000 abstract/normalized-list pairs for three different organisms: fly, mouse and yeast. They also provided extensive synonym lists for all three organisms.

It should be noted that gene normalization is both easier and harder than identifying gene mentions. It is easier because it does not require textual boundaries of each mention to be identified, but only that some mention be detected and the document annotated accordingly. On the other hand, gene normalization is harder than identifying mentions in that it requires the actual gene to be identified. The three organisms under consideration, yeast, fly, mouse, have from thousands to tens of thousands of genes. Even if it were possible to identify every gene mention with 100% accuracy, it would still be difficult to disambiguate each mention given the number of possibilities and the high degree of overlap among synonym lists for different but related genes.

Normalized Form	Possible Synonyms
FBgn0003943	CG11624 Ub, Ubi p, Ubi63E, polyubiquitin
FBgn0003944	CG10388 Cbx, DmUbx, Hm, Ubx, abx, bithorax
FBgn0003945	Udg, Uracil DNA glycosylase

Table 1: Example entries in the fly synonym list.

Section 2 describes some initial directions we considered. Section 3 describes a pattern-matching approach, and Section 4 a machine-learning approach based on maximum-entropy classification. Section 5 compares the two methods.

2 Initial Directions

With the availability of highly accurate gene taggers [7], one simple approach would be to extract all the gene mentions from text and to match these mentions to the synonym list of each organism. However, there are many difficulties with that approach. The primary problem is that mentions may be ambiguous. For instance, the gene mention *alcohol dehydrogenease* is a valid synonym for 111 different genes for the fly organism. Simply matching *alcohol dehydrogenease* to all 111 genes would lead to a steep decline in precision (since the mention is most likely referring to only one specific gene). Second, the system would be reliant on the accuracy of the gene tagger. Our experiments showed that for mouse, the gene tagger performed reasonably well on the development data. However, for fly and yeast, the tagger's performance was less than useful. This is most likely a result of the fact that the gene taggers training data did not contain a sufficient amount of examples for those organisms.

Another seemingly straightforward approach is to treat the problem as multi-class document classification. Here, each normalized gene form is a possible classification and the goal is to classify each document with some set of genes. We encountered two major problems to this approach. Multi-class document classification is typically done for tens and in rare instances hundreds of classes. However, as stated earlier, each organism has thousands of genes and in some cases tens of thousands. This poses substantial computation issues. Another major obstacle is that not all classes are observed in the training data. Only 22%, 13% and 47% of all fly, mouse and yeast genes are ever seen in the training data. This would make it impossible to gather the sufficient statistics needed to make accurate predictions.

3 Pattern Matching

Given a synonym list that is both unambiguous and exhaustive, creating a normalized gene list would be simple. We could simply match every occurrence of a synonym in the text, and based on those matches label the document with the corresponding normalized mentions. Unfortunately, the synonym list for this task has neither of the properties we desire. As previously mentioned, many synonyms are ambiguous, either occurring with multiple genes or in contexts where no gene mentions are present. But even with these ambiguities, which increase greatly the number of genes that a simple pattern matching system would propose, using the synonym list doesn't retrieve all mentions.

This does not mean that a pattern matching approach is useless — our first system relies heavily on standard techniques. However, we do not assume that every synonym in the list reliably labels documents with their normalized gene mentions. Instead, we prune each organism's synonym list so that it only contains synonyms that we believe will be informative, based on labeled training documents. A synonym, s, for a gene, g, is considered informative if and only if for the training set D:

$$\frac{\sum_{d \in D} \text{match}(s,d) \times \text{labels}(g,d)}{\sum_{d \in D} \text{match}(s,d)} > \delta$$

where $\text{match}(s,d) = 1$ if there is an exact match of s in document d and 0 otherwise, and $\text{labels}(g,d) = 1$ if g is a member of document d's gene list and 0 otherwise. The left-hand-side fraction is the conditional probability of g labeling a document, given that there was a match of s in the document. The threshold δ was tuned to 0.4 on the development data set.

While using a pruned synonym list performs significantly better than simple pattern matching with the original list, we still predict far too many genes for each document. To further restrict the genes considered, a second stage of the pattern matching system produces, for each document, a set of candidate genes. Now, only genes that are present in the candidate list for a document and are also associated with an informative synonym in that document will be added to the document's final list.

For a document in the fly organism, the system extracts the 1000 closest documents in the training data using a standard cosine distance metric. The gene lists for the neighbouring documents are merged to create the candidate list. For a document in the mouse data, the system first tags the document using a gene tagger [7]. Each gene mention is then compared to every synonym in the mouse synonym list. If the gene mention and a synonym have a Jaro-Winkler similarity [4] greater than 0.85, then the gene that synonym is associated with is added to the candidate list for that document.

fly	**Precision**	**Recall**	**F-measure**
basic matching	0.033	0.861	0.063
informative syns	0.458	0.727	0.562
candidate list	0.709	0.667	0.687
stemming	0.713	0.690	0.701
mouse	**Precision**	**Recall**	**F-measure**
basic matching	0.151	0.583	0.240
informative syns	0.478	0.548	0.511
candidate list	0.739	0.505	0.600
stemming	0.716	0.656	0.685

Table 2: Precision, recall and f-measure for the pattern matching systems on devel. data. Each system is a superset of the previous.

Organism	**Precision**	**Recall**	**F-measure**
fly	0.638	0.695	0.665
mouse	0.830	0.673	0.743
yeast	0.950	0.894	0.921

Table 3: Precision, recall and f-measure for eval. data with complete pattern match system.

This two-stage pattern matching system compensates for the fact that the given synonym list contains large amounts of ambiguity, but does nothing to reduce the number of gene mentions that a naive pattern matching approach misses. We observe that many of these omissions occur because of differences in punctuation or morphology. Thus, the pattern matching system includes a third, and final stage. In it, all punctuation is removed and each token is stemmed with the Porter stemmer [9], in both the documents and the synonym lists. As before, each informative synonym is compared to each document. If the synonym matches and the corresponding gene is in the candidate list for that document, then the gene is added to that document's final gene list.

Finally we will note that the yeast system required neither a candidate list nor stemming to gain maximum performance.

Table 2 shows the performance for the different stages of the system for the fly and mouse organisms. Table 3 summarizes the final results of the system on the evaluation data.

3.1 Shortcomings of Pattern Matching

It is rather surprising how well pattern matching can do when one is smart about it. However, there is something unsettling about this approach. First, several parameters need to be adjusted on the development data. These include δ as well as the various parameters required to create the neighbour lists (i.e. number of closest documents and the Jaro-Winkler distance). For all hand-tuned parameters there is the danger of overfitting to the development

data.

The complexity of the system is also a problem. Data is transformed in many stages: stemming, creating the informative synonym list, creating the neighbour list and finally matching the synonyms to the text. As with all pipelined systems this may lead to cascading errors in which an error early in the pipeline will cause errors to be made at later stages.

A lack of uniformity between each organisms system is also an undesirable trait. Particularly, neighbour lists are generated differently for each organism or not at all in the case of yeast. One could easily argue that the method will not generalize well to other organisms. What we really desire is one uniform approach, for all organisms, in which every parameter is automatically set during the training phase.

4 Match Classification

The inspiration for our second model comes from the observation that liberal pattern matching from the synonym list to the document can achieve a very high recall (91%, 79% and 90% for fly, mouse and yeast on the development data). The problem, as addressed in the last section, is that this also results in extremely poor precision. However, just as it is possible to use the training data to determine which synonyms are useful, it is also possible to use the training data to determine which matches are good and which are bad.

We present here a model that, given a set of synonym matches, distinguishes those that are good from those that are bad. This is essentially a binary classifier in which good matches are positively labeled and bad matches negatively labeled. To create training data for the classifiers, we matched every synonym to each training document using a loose matching criteria (punctuation and numbers were ignored). We then extracted for each match, the text that matched, some context of the match, the normalized form causing the match, as well as the number of other genes which matched that specific piece of text. For the training data, if the normal form for a match was in the normalized gene list for that document, then the match was labeled positive. Below are three example matches:

of drosophila *Kinesin heavy chain* attached to, FBgn0001308, 1, Y
was analyzed *in* trajectories with, FBgn0001250, 5, N
homeotic gene *Ultrabithorax* (ubx, FBgn0013100, 7, N

The italicized text is the text causing the match. We extract two words before and after the match. In the first example, the normalized form causing the match is FBgn0001308, it was the only gene matching that piece of text and it constituted an actual match. Note that the third match, *Utrabithorax*, is negative because it is actually a match for the gene FBgn0003944, which shares the synonym *Utrabithorax* with FBgn0013100.

Organism	Precision	Recall	F-measure
fly	0.704	0.784	0.742
mouse	0.787	0.731	0.758
yeast	0.956	0.881	0.917

Table 4: Precision, recall and f-measure for eval. data with match classification model.

This provided a large set of positive and negative matches required to train a classifier. We used the MALLET [6] implementation of maximum entropy models [2] for our classifiers. Maximum entropy classifiers model the conditional probability of a class given an input vector with the log-linear form:

$$P(y|\mathbf{x}) = \frac{e^{\sum_i \lambda_i f_i(y,\mathbf{x})}}{Z(\mathbf{x})}$$

where y is a class (in our case Y or N), $\mathbf{x}$ is an input vector and $Z(\mathbf{x})$ is a normalizing term. For our model $\mathbf{x}$ is a binary vector containing predicates on the matched text, its context, the normal form causing the match and the number of other genes matching the text. Each feature function $f_i(y, \mathbf{x})$ maps an input vector and class to to a binary variable, for instance:

$$f_k(y, \mathbf{x}) = \begin{cases} 1 \text{ if ContextBefore=drosophilia is true in } \mathbf{x} \\ \qquad\qquad\qquad\qquad \& \ y = \text{Y}; \\ 0 \text{ otherwise.} \end{cases}$$

The parameters of the model are the feature weights λ_i. Ideally one would like the weights of features that tend to be on for correct classifications to be strongly positive, the weights of features that tend on for incorrect classifications to be strongly negative, and the weights of uninformative features to be zero. To accomplish this the parameters are set to maximize the log-likelihood of the training data $\mathcal{T}$:

$$\mathcal{L}(\mathcal{T}) = \sum_{(y,\mathbf{x})\in\mathcal{T}} \log P(y|\mathbf{x})$$

A Gaussian prior over weights, with variance tuned to 1.0 on the development data, reduces the danger of overfitting [3]. Optimal parameter values are found by numerical optimization using a limited-memory quasi-Newton methods, which guarantees convergence to a global maximum for concave functions like the penalized log-likelihood we use.

Results of the trained models on evaluation data are shown in Table 4.[1]

[1]The evaluation results differ from the official BioCreative 2004 results. This is due to a match extraction error that was discovered and resolved after the official results were submitted.

5 Results and Discussion

Comparing Table 3 and Table 4, we see that maximum entropy classification does just as well or better than the pattern matching system described in Section 3. A primary advantage of maximum entropy classification over pattern matching is that the system is uniform across organisms, hence the method is more likely to perform well when extended to different organisms.

There are many ways in which the maximum entropy model can also be improved. The most obvious of which is to include more expert knowledge into the model. Maximum entropy models are widely used since they easily allow for the integration of such expert knowledge through the definition of new features. Currently, the model's features are based primarily on textual matching and contain no domain specific information. Another potential improvement would be to relax the criteria when extracting matches. Under perfect conditions we should be able to extract all good matches and use the classifier to eliminate the bad ones. Currently our matching criteria extracts as low as 80% of all good matches, which bounds the recall of the system. We are experimenting with different string distance metrics proposed by Cohen et al. [4] to try and raise the number of good matches returned.

Acknowledgments

The authors would like to thank Mark Liberman, Mark Mandel, Andy Schein, Scott Winters and Pete White for useful discussions and guidance. We are also very appreciative of Andrew McCallum for making an early version of MALLET available to us.

References

[1] A critical assessment of text mining methods in molecular biology workshop, 2004.
http://www.pdg.cnb.uam.es/BioLINK/workshop_BioCreative_04/

[2] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1), 1996. 2001.

[3] S. F. Chen and R. Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99- 108, Carnegie Mellon University, 1999.

[4] W. Cohen, P. Ravikumar & S. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks, in *IIWeb workshop*, 2003.

[5] J. Kazama, T. Makino, Y. Ohta and J. Tsujii. Tuning Support Vector Machines for Biomedical Named Entity Recognition. In *the Proceedings of the Natural Language Processing in the Biomedical Domain*, ACL, 2002.

[6] A. K. McCallum. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu 2002.

[7] R. T. McDonald and F. Pereria. Identifying gene mentions in text using conditional random fields. In *A critical assessment of text mining methods in molecular biology workshop*, 2004.

[8] M. Narayanaswamy, K. E. Ravikumar, K. Vijay-Shanker: A Biological Named Entity Recognizer. *Pacific Symposium on Biocomputing*, 2003.

[9] Porter, M.F. An algorithm for suffix stripping, *Program*, 14(3): 130-137, 1980.