Finding biologically relevant protein information using an unsupervised learner of natural languages

ThaiBinh Luong
CBB545
Final Project
5/7/07

1 Introduction and Background Information

Text mining in the biomedical field is useful from the perspective of both computer science and biology. For text miners, the vast amount of freely available text from one source (PubMed[1]) enables testing on a large corpus. For biology researchers, text mining provides a possible alternative to having to read each article that is published. As many papers come out each week, there is often a lag period between discovery and widespread recognition simply because researchers cannot keep up with all the literature, especially if a finding is in an obscure journal.

The key application of data mining with respect to biomedical literature is to find underlying relationships that come from pooling a number of abstracts together, rather than from just one abstract alone. Take the following example. Suppose there are two abstracts, that each provide us with the following bits of information:

Abstract 1: ProteinA inhibits ProteinB
Abstract 2: ProteinB activates ProteinC

Using the knowledge from these two abstracts together, we can infer that ProteinA indirectly inhibits ProteinC. This specific example of a relationship is commonly referred to as a protein-protein interaction.

Protein-protein interactions are among the hardest information to extract from biomedical literature. The task of finding interactions can be split up into a number of smaller subtasks, such as finding relevant articles, finding relevant sentences within those articles, and determining which proteins are involved in the interaction. In a recent combined effort to compare text mining systems in a consistent fashion, one of the subtasks in the BioCreative challege[2] was to find the pairs of proteins involved in a protein-protein interaction. Out of the 16 teams that participated, no team achieved an f-score of greater than 0.288, and some teams scored in the single digits.

The reason why text miners pursue protein-protein interactions is to establish full pathways (i.e. a network of interactions) for proteins. A fully understood pathway can aid in drug targeting, because researchers would have a number of possible proteins at which to aim drugs, and they can predict how a drug may affect other proteins.

My project is essentially to use PubMed abstracts to extract information about proteins by determining how they regulate each other.[*]

2 (Main) Tools Used

2.1 ADIOS – Automatic distillation of structure

ADIOS[3] is the main driving force behind this project. It uses unsupervised learning of natural languages to construct a grammar. In text mining, this is especially appealing because unsupervised learning requires minimal prior knowledge of the language; A tagged training set is not needed, so this approach saves a lot of resources.

The input for ADIOS is a corpus of sentences, and the output is a grammar. ADIOS is run in the Linux environment. An additional note is that the demo version of this program only allows 100 rules to be constructed.

2.2 Abner – A Biomedical Named Entity Recognizer

Abner[4] is a named entity recognizer, which means that given a string of text (in our case, an abstract taken from PubMed), it tries to recognize entities that represent proteins. Although this seems like a trivial task, there are many factors that make this task more complex; e.g. New

[*] Genes are sections of DNA, and they are used as a template for creating proteins. Proteins are the actual molecules that carry out the work. Generally, both the gene and the protein encoded by that gene have the same name, so I use “gene” and “protein” almost interchangeably throughout this report.

gene names are created all the time, and authors aren’t consistent in naming genes/proteins in their articles.

The overall recall and precision of proteins using Abner is 77.8% and 68.1%, respectively. One of the reasons Abner was used is because it has a wrapper written in Java. Many other biomedical-related text mining tools are web-based applications.

2.3 Programming Languages

Other than using Java with Abner, most files were parsed and prepared using Python. To visualize the grammars created by ADIOS, a small script was written in MatLab.

3 Methods

3.1 Attempt #1

I started with a list of 11 keywords that I felt would appear in abstracts discussing how one gene affects another gene:

`regulates`	`promotes`	`stimulate`
`inhibits`	`expression`	`upregulation`
`transcription`	`express`	`downregulation`
`activates`	`antagonize`	

I did a pubmed search for each word, and retrieved the first 200 abstracts returned from that search. Removing repeats, there were 1,866 abstracts, which resulted in 22,205 sentences.

Using Abner, I found the genes that were mentioned in each of the sentences, and only kept the sentence if it had more than one gene mention. This reduced the set of sentences to 5,224 sentences. In each sentence, the actual gene name was replaced by the generic term “Gene#”, where ‘# = 1’ if that gene was the first to appear in the sentence, ‘# = 2’ if the gene appeared second in the sentence, etc. After converting the sentences into the correct ADIOS format, the input would look like this:

```
*  Phosphorylated Gene1, one of the components of Gene2, was also detected  #
*  Gene1 expressions in KF following various dosages of Gene2 were assessed  #
*  Gene1 Activates Gene2 and Modulates Cell Cycle Proteins in OKP Cells  #
. . .
```

I defined the training set as 80% of the total sentences, or 4,179 sentences.

After running ADIOS on the training set, although the system was able to create a grammar, rules were not able to generalize gene interactions well because the sentences varied too much. It mainly gave rules for parts of the sentence that would lead into a relevant finding. Because the constructed rules are based on the order of the input, I tried scrambling the training sentences, but subsequent runs did not yield any better results. An example is shown in figure 1.

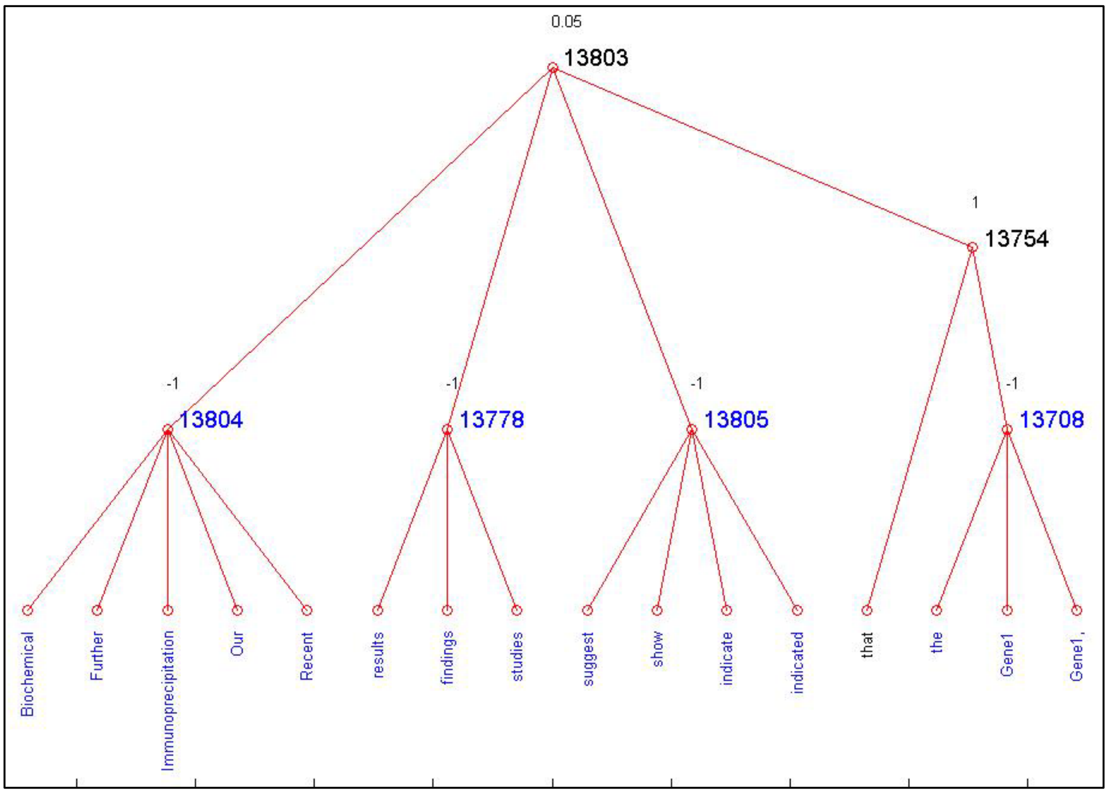


Figure 1

Patterns are represented by the nodes labeled with black number, while equivalence classes are represented by blue nodes.

3.2 Attempt #2

In the first attempt, I noticed that a lot of the rules only summarized the parts of the sentence that led up to, but did not include the actual gene interaction, such as “Our study finds that…”. Generally, the title of an article summarizes the entire abstract succinctly, so I decided to use the titles instead of the sentences within the abstract. Although there were initially 1,866 abstracts (and therefore the same number of titles), that number dwindled to 267, after saving only those titles with co-occurrences of genes. This small group of titles was not enough to create *any* patterns whatsoever.

3.3 Attempt #3

In my third attempt, I tried a completely different approach altogether. The National Cancer Institute has a “Cancer Gene Data Curation Project”[5], in which one of the curated databases contains associations between genes and diseases. This database was downloaded as a flatfile, and required a small amount of clean-up for consistency (to aid in the automatic parsing of the file). This database contained 1002 unique genes. I used these genes as a starting point (rather than the list of keywords from above).

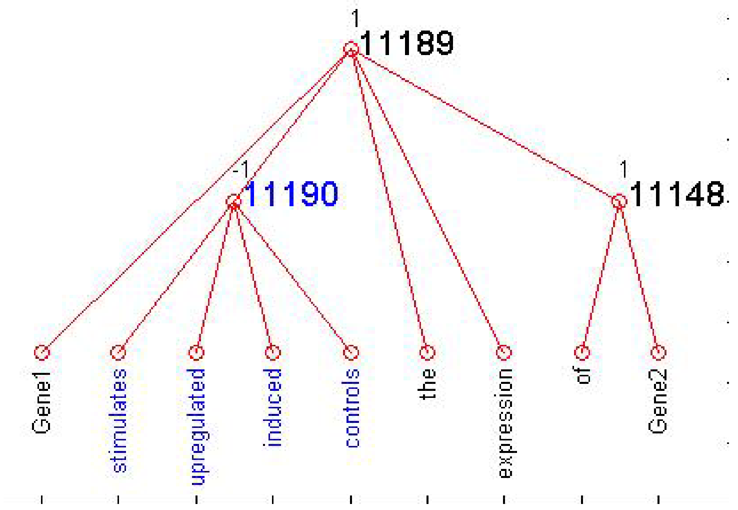
My eventual goal was to use two different tools: LitMiner[6], and MedMiner[7]. LitMiner takes one gene as an input, and returns a list of most co-cited genes. The input for MedMiner is two genes, and it returns abstracts and relevant sentences that co-cite the two genes. My plan was to use those output sentences as training sentences for ADIOS.

Even starting with the first tool, LitMiner did not find many gene co-occurrences with the cancer-related genes taken from the database. In some cases in which co-occurrences were found, MedMiner was not able to find the corresponding abstracts/relevant sentences. In addition, these two tools are web-based, so I could not run batch files. Consequently, gene names were copied/pasted one at a time into the query boxes, and the output was in html format. This method of obtaining sentences did not look promising, so I dropped it and returned to my original set of sentences (from Attempt #1).

3.4 Attempt #4

Using the 5,224 sentences from Attempt #1 (which each contain at least 2 gene mentions), I chose sentences which also mentioned one of the stemmed forms of the words in the keyword list. For example, I included sentences that had any form of ‘regulat-‘, such as “regulate”, “regulates, “upregulating”, etc. Using this strategy, I was left with 3,237 sentences (including 2,590 training sentences). That training size is actually ideal for the ADIOS demo version, because the demo version only allows up to 100 patterns. From running earlier training sets, I observed that the upper limit is reached around sentence number 2,000.

This attempt finally yielded decent results. Here is an example pattern:



To automatically extract gene regulation information from other, novel sentences, I would “tag” equivalence class 11190 with “activates”, because each of the terms in that class refer to activating Gene 2.

4 Results and Concluding Points

Despite slightly better grammar rules, the system was still not able to recognize *any* of the sentences in the test set. I even added in two, very basic sentences, but ADIOS did not match them to any of the existing patterns. This could possibly be a limitation to the demo version, as the number of rules were restricted. This would make each input sentence much more valuable, so careful selection for the input is required.

I would have liked to try different approaches to this problem, as I did in Approach #3. Although many tools have been developed specifically for biomedical knowledge discovery, a lot of them are based online. Usually, a biologist has one favorite gene that he/she is researching, and will do a specialized search revolving around that one gene. For this reason, it is hard to do batch file data mining.

5 References

[1] http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=PubMed
[2] Krallinger, M., Leitner, F., Valencia, A., Assessment of the Second BioCreative PPI Task: Automatic Extraction of Protein-Protein Interactions. *from Proceedings of the Second BioCreative Challenge Evaluation Workshop*. Madrid, Spain, 2007.
[3] Solan, et al., Unsupervised learning of natural languages. *PNAS* vol. 102, August 2005.
[4] Settles, B., ABNER: an open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191-3192., 2005.
[5] http://ncicb.nci.nih.gov/NCICB/projects/cgdcp
[6] http://andromeda.gsf.de/litminer?choice=genomdb
[7] http://discover.nci.nih.gov/textmining/entry.jsp