CPSC 490 Project Proposal: Biological Entity and Relationship Extraction

Robert Tung and Adrian Lin

Advisor: Dragomir Radev

Submitted: September 21, 2017

1 Abstract (Verbatim from Project Proposal)

Millions of medical research papers have been written in the past century. This explosion in research publications has created an mountain of information that can be difficult to process by a team of researchers much less a single person.

We seek to use natural language processing methodologies to create a tool that can take in one or more biological papers and extract, with high confidence and minimal false positives, biological entities (e.g. genes, mutations, diseases, etc.) and their relationships. Specifically, this project will look at the effect of genes and mutations on diseases.

The tool will be incorporated into a web application and should also be able to extract overall deductions from an aggregate set of papers. If time allows, the strength of extracted relationships should be measured using a confidence metric.

2 Project Outline (Verbatim from Project Proposal)

Overall, this project will require us to do the following

1. Scrape for Medical Papers - figure out which papers need to be downloaded and build up a corpus
2. Find a PDF to Text Parser and clean up the resulting text.
3. Use named-entity extractors to recognize entities such as proteins and diseases. Find the best named-entity extractor and tune or modify. If necessary, create our own named-entity extractor for this specific area.
4. Find a library that extracts relationships between entities in text. Find the best relationship extractor and tune or modify. If necessary, create our own relationship extractor for this specific area as well. This may be likely since the quality of existing relationship extractors is not as high.

5. Create a web application that takes in a paper and returns the biological entities and relationships.

6. Stretch Goal: Track the source (provenance) of each relationship and update the confidence of each relationship.

7. Stretch Goal: Create a more accurate confidence measure for each relationship extracted. Use this to have aggregate information over a set of papers.

8. Stretch Goal: Update the web application to take a set of papers and extract overall deductions and aggregations. For example, these set of 30 papers found that gene X is correlated with disease A while another 20 claim that gene Y is correlated with disease A.

3 Work Completed (Adrian)

I have built up a suitable and sizable corpus of medical papers. The PMC (PubMed Central) Open Access Subset is a corpus of articles which are available to the public and not subject to traditional copyright laws. These papers fall into two categories of commercial and non-commercial use and are available for bulk downloading in both txt and xml forms. These papers are further partitioned into their respective categories (e.g. Advanced Bioinformatics, Cardiovascular Disorders, etc). Rather than blindly randomly sampling from these papers, I instead used papers which categories begin with the letters A or B, for a total of around 110,000 papers. These papers in XML format are now being stored on the tangra servers. Using scripts that Robert has written, these papers can be converted into plain text quite easily. Furthermore, I have also written a script to traverse through all directories and extract only the abstracts from each paper and write them to a single file. We may ultimately want to use only the abstracts from papers especially in training our models for entity and relation extraction.

Next, I began work finding libraries for relation extraction. The library desired must fulfill a number of criteria. It must be able to take in a custom corpus of text and tags as generated by our named entity extractors. Most generic out of the box relation extraction libraries have pre-existing categories for named entities such as location, person, organization, etc. However, our project will be working specifically with genes and diseases and not much outside of this. NLTK's relation extraction library relextract has 3 built-in corpora, ieer, conll2002, ace. Using relextract, I have written a script that given a type of relation and two entity categories, parses a text and returns all instances which match the relation and entity type. One major limitation of NLTK's relextract is that its relation extractions are based on regular expression matching. I do not think that regex-based relation extraction will be sufficient to fully capture the causation or correlation we want to mine.

4 Work to be Done (Adrian)

Relation extraction tools and libraries are much less well defined than those for named entity extraction. The bulk of my remaining work will be to find the necessary tools for linking genes entities and disease entities. NLTK's regex-based tools will not be sufficient to complete the task. What we will likely end up using is a co-occurence based extractor such as [Chun et al., 2006] or a syntactic based extractor such as [Quan and Ren, 2014]. In addition, pre-existing tools such as DisGeNET [Dis,] and Chilibot [Chi,] may provide useful datasets for further training. A main component of the work will be finding a large enough corpus to train on. Engineers at Microsoft have done similar tasks using distant supervision. [Mic,]. One existing tool which shows promise is the DISEASES database for genes and disease relations [Pletscher-Frankild et al., 2015]. This tool uses a strict co-occurence model, meaning that it detects a strong relation between a gene and disease if they are mentioned in the same abstract. This approach does not use any natural language processing techniques. Nevertheless, these abstracts can be potentially scraped for training data and parsed into dependency trees for more accurate representation.

Finally, I will link the finished relation extractor to our web app which is built in Flask and can run Python scripts directly. If all the above is finished, I will move on to developing metrics to measure the strength of gene-disease relationships. This may involve sentiment analysis-like work on the actual text to determine the certainty of the researchers' claims.

5 Work Completed (Robert)

To begin, while Adrian found the corpus of medical papers in xml format, I wrote a python script that takes an .xml file and extracts the text of the paper (ignoring extraneous xml tags and the like) and writes this out to a .txt file of the same name. I tested this script on papers that Adrian found and confirmed that it works.

After completing the above, I began to work on the named-entity extractor. So far, I have completed the implementation of the named-entity extractor for genes, and am beginning work on the one for genetic diseases. The named-entity extractor for genes is built from nltk and uses a Naive Bayes Classifier to train. Currently it is based on the syntax of gene names, since the syntax seems to be strictly enforced [gen, b].

Building the above named-entity extractor for genes also required me to compile an exhaustive corpus of genes. I began by Googling for lists of most important or influential human genes (which also required me to script the extraction of these gene names from the html of the sites that I found them on) and writing a python script to find the unique names among this compiled list from several sites [wik, , emp, , gen, a, ide,]. Later, I also found a site [Andrew Yates, 2016] with a (presumably comprehensive) list of over known 60000 human genes and names, and have included this file as well. Training on this file takes a bit longer than the previous file, but it performs a bit better as well.

I also determined which features to use to train for gene names. I considered testing whether a word was in the comprehensive gene corpus, but this would take quite a while. In particular, this operation alone would be $O(mn)$ where m is the size of the corpus of genes and n is the length of the paper. Thus, instead of this, I instead found features about the structure of the name itself, in particular the length of the name, the proportion of the word which is capital letters, the proportion of the word which is numerals, and whether the word was a dictionary word (using the enchant library for dictionary-checking).

The above named-entity extractor was trained on the list of human genes and given the text of "Sense and Sensibility" as an arbitrary, long, text of non-genes. In doing so, it was able to label $\frac{60947}{63967}$ genes correctly (95% accuracy) and it was able to label $\frac{134955}{135412}$ non-genes correctly (99.66% accuracy) for a second text of non-genes (A Tale of Two Cities).

I also have a separate script for reference that uses an nltk Tree structure to more accurately extract named entities, but which has not been customized to classify genes. This code could be modified in the coming weeks to improve the above entity-extractor.

Finally, I began creating the Web-app which will eventually be able to take a file and perform our named-entity and relation extraction. It is currently a working Flask app that can take a file input on the UI and, for the sake of testing the read functionality, print the file back out.

6 Work to be Done (Robert)

The above named-entity extractor could be tuned further and I can still try other libraries / stacks to see if better performance can be reached.

More imminently, I will continue work on the named-entity extractors for genetic diseases, along with any other biological entities that we deem relevant to this project. One hurdle with this is that I should be able to find entities that are multiple words, as most diseases are. For this, the extra code I mentioned above that uses nltk Trees may come in handy. Additionally, while genes have a specific syntactic structure, diseases are not identifiable syntactically but also require semantic information.

After doing the above, we met with Professor Radev and determined that the best way to approach the named-entity extraction of diseases (and possibly genes in fact) is to look into existing work; we can both adapt and modify these existing tools and create original tools that are compatible with the same libraries. In particular, the NCBI Disease Corpus provides training, validation, and testing sets of text with tagged diseases [DoÄŸan et al., 2014]. Moreover, Sunil Sahu and Ashish Anand created RNN models for disease name recognition [Sahu and Anand, 2016] and, while the code is not given in the paper, this is also worth looking into. Additionally, Abner is a biomedical named entity recognizer [Settles, 2005] with a Java API that may be useful to us. Similarly, becas is a biological concept annotation tool with a Python

API that could be very relevant [Oliveira, 2013].

After adapting existing tools and creating our own, I will also perform error analysis to tune and determine the best tools to use. I will compare these to the error / accuracy of the existing literature.

Once this and Adrian's relationship-extractor are completed, those functionalities can be linked to the webapp (which should be fairly intuitive as the webapp itself is built in Flask and thus can access and use other Python files).

If all of the above is done and with suitable accuracy, we can then move onto our stretch goals. For this, I can begin tracking and storing the source of each relationship found and use machine learning with this data to inform the confidence of the relationship, and use this to allow the web-app to only return relationships with high confidence. We can also incorporate features of the source (e.g. author, verbosity of the paper, length of the paper, etc.) to inform this confidence.

References

[emp,] The 20 most studied genes.

[Chi,] Chilibot: finding gene and protein relationships from medline - a text mining approach.

[Dis,] Disgenet - a database of gene-disease associations.

[gen, a] Genage human genes: List of entries.

[gen, b] How are genetic conditions and genes named? - genetics.

[wik,] List of human genes.

[ide,] The top 10 human genes.

[Mic,] Training a classifier for relation extraction from medical literature.

[Andrew Yates, 2016] Andrew Yates, Wasiu Akanni, M. R. A. D. B. K. B. D. C.-S. C. C. P. C. S. F. L. G. C. G. G. L. G. T. H. S. E. H. S. H. J. N. J. T. J. S. K. I. L. F. J. M. T. M. W. M. D. N. M. R. N. M. N. A. P. M. P. M. P. M. R. H. S. R. D. S. K. T. A. T. A. V. S. P. W. A. Z. E. B. J. H. M. M. E. P. M. R. G. S. S. J. T. F. C. B. L. A. D. R. Z. P. F. (2016). Ensembl 2016. *Nucleic Acids Res.*

[Chun et al., 2006] Chun, H.-W., Tsuruoka, Y., Kim, J.-D., Shiba, R., Naoki, N., Hishiki, T., and Tsujii, J. (2006). Extraction of gene-disease relations from medline using domain dictionaries and machine learning. 11:4–15.

[DoÄŸan et al., 2014] DoÄŸan, R. I., Leaman, R., and Lu, Z. (2014). {NCBI} disease corpus: A resource for disease name recognition and concept normalization. *Journal of Biomedical Informatics*, (0):–.

[Oliveira, 2013] Oliveira, T. N. D. C. S. M. J. L. (2013). Becas: biomedical concept recognition services and visualization. *Bioinformatics*.

[Pletscher-Frankild et al., 2015] Pletscher-Frankild, S., Pallejà, A., Tsafou, K., Binder, J. X., and Jensen, L. J. (2015). Diseases: Text mining and data integration of disease–gene associations. *Methods*, 74(Supplement C):83 – 89. Text mining of biomedical literature.

[Quan and Ren, 2014] Quan, C. and Ren, F. (2014). Gene–disease association extraction by text mining and network analysis.

[Sahu and Anand, 2016] Sahu, S. K. and Anand, A. (2016). Recurrent neural network models for disease name recognition using domain invariant features. *CoRR*, abs/1606.09371.

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