Information Extraction from Biomedical Texts

Nathan Srirama, Zherui Lin

1. Introduction

Biomedical research has been identified as a very important area of research in the 21st Century. The New Hampshire IDEA Network for Biomedical Research Excellence noted, “Everybody in America has enjoyed the benefits of biomedical research. From the creation of new vaccines, drugs, or procedures to treat or prevent illnesses, to the safety testing of items we use in our day-to-day lives, biomedical researchers strive to better comprehend the causes and cure of illnesses” [1]. In addition, in the digital age, current biomedical research has become increasingly data-driven. Researchers lean more and more on computer automated processes to create simulations and accelerate research efforts. A large part of this involves identifying protein and gene names within bodies of text for analysis. These names are currently mostly hand-tagged by researchers for computer use. Our goal was to create a way to automate this process of identifying protein and gene names within biomedical texts. In addition, we utilized a dataset of extracted sentences containing potential protein names and performed a hierarchical clustering on those sentences to look for patterns in sentence structure that could be used to create generalizable rules for future protein identification methods.

2. Related work

The work in this paper was largely inspired by the work done by Kuo and Lin from the National Chiayi University in Taiwan. Their work focused largely on utilizing a mixture of heuristics to extract protein names from biological literature [2]. The method they created for identifying protein names within biomedical text involved a combination of rule based. The specific tool they used in their research was regular expression. They developed 7

categories rules to extract protein names for biological literature.

The other technique we implanted was Name Entity Recognition. Named Entity Recognition (NER) is a method to automatically identify named entities in a text and classify them into predefined categories. It is often used for categorization and content recommendation. An example of NER would be: “Mark Zuckerberg is one of the founders of Facebook, a company from the United States.” There are three predefined categories in this sentence: person (Mark Zuckerberg), company (Facebook), and location (United States). Through NER, we hope to capture the pattern of the sentences that contain protein names. The pattern of the sentence can be analyzed with the idea of clustering.

Cluster Analysis is the task of taking a set of n-dimensional data and identifying how similar or different certain datapoints are to each other. This is a common technique used in pattern analysis, image processing, data compression, and machine learning [3]. The goal is to calculate a measure of distance between all points within a dataset. This measure of distance can be generated using any distance algorithm, including Euclidean, Manhattan, Minkowski, or any other pre-defined distance formula. A lot of program implementations of cluster algorithms also let you define unique distance functions for specialized purposes. Hierarchical clustering is a type of cluster analysis that focuses on creating a hierarchy of clusters, where clusters are defined at different levels based on how similar points are to each other. Very similar points will have a cluster at a low level of the hierarchy, and at the very top will be the entire dataset encompassed in one massive cluster.

3. Methodology

This project is split into two distinct parts. The first involves utilizing some

database and rule-based methods to identify sentences that are likely to contain protein names. The second utilizes the data that was identified from the previous step in order to do cluster analysis on sentence structures, with the goal of identifying patterns that may be helpful for further protein name extraction from texts. In addition, we include a description of our main dataset for clarity.

3.1: Dataset

The literature we used to extract the protein names is “Innate Immunity to Respiratory Infection in Early Life” [4]. It consists of 3630 words and 130 sentences. It is relatively small for NLP research, but the goal for this project is to demonstrate the idea of information extraction as a proof of concept. It was not deemed necessary to use a bigger sample.

The research of protein names required expert experience in a related field. Therefore, we think it is appropriate to use a database from official and trustworthy sources. The UniProt consortium comprises many research institutes from Europe and is based in Washington DC. The uniprot[5] database is special for biological related research and contains many proteins information from biological literature. The database is constantly updated and free to access. The database for protein names is large and takes a considerable amount of time to download. So, we subset the dataset and only looked at proteins associated with humans. This came to about 20000 entries.

3.2: Heuristical Protein Recognition

3.2.1 Protein Name Extraction

Followint the idea of protein names extraction by Kuo and Lin [2], the first problem we need to solve is to identify the protein names. In their work, they categorized all the words into seven classes. These classes are delimiter, single, abbreviation,bioregular,regular,sequence,

other. Even specific regular expressions for each class are not perfect and do not always correspond to a protein name, but it is still valuable for us to filter out some protein names.

3.2.2 Dimension Reduction

Once we extracted the protein names, we noticed there are two problems with the words we extracted from the literature. The first one is the repetition of words. Since we did not limit the occurrence of words, it is expected that some of the words appear more than once. Therefore, we removed the duplicated words through the Python list(set) function [6]. Originally, we have 1220 words and cut them down to 987 words.

The other problem of the extraction was unrelated words. As we mentioned earlier, the regular expression we implemented is not perfect. Hence, we need to find a way to eliminate unrelated words. NLTK is a python platform special for language processing and it offers more than 50 corpora. The brown corpus [7] contains words from adventure, religion, fiction, and other fields which do not relate to biology. Once we cross-check our words list with the brown corpus, we are left with 275 words.

3.2.3 Database examination

After we downsize our list to a reasonable size, we meant to compare it with the Uniprot database. However, the characteristic of protein names gives another problem. Unlike other objects that we are familiar with. The protein names usually consist of more than one word. For example: TBC1 domain family member 9B. Only a limited amount of protein is less than two words. Therefore, we have two approaches to solve this problem.

The first approach is to limit the length of the protein name and only match the names with less than two words. By doing this, we can guarantee that the words we matched are highly related to proteins.

However, given the size of the database and literature, we did not find any match.

So, we took the second approach which is to match the database regardless of the length of the name. The risk of this approach is we will end up with many names which do not relate to proteins. Since there was no certain expectation of accuracy, this is a harmless action and at least a match is guaranteed. In the end, a total of 16 words considers relating to proteins.

Once the words are isolated, we put them back to the biological literature and find the sentence that matches the word. Examples will present in the result Section 4.1.

3.3: Pattern Analysis utilizing Hierarchical Clustering

Pattern analysis can be done on sentences that contain potential protein names in order to identify similarities in their sentence structures. These similarities could then be applied in order to help identify more sentences that contain protein names. This pattern analysis was accomplished utilizing hierarchical clustering. In order to apply clustering algorithms to our dataset we must formulate the format our data will exist in, as well as how the distance between two data points will be defined.

3.3.1 Pattern Analysis Data Format

The Parts of Speech (POS) of words was identified as an adequate way by which we could analyze sentences for patterns regarding protein names. Python's NLTK package defines a total of thirty-one parts of speech. These parts of speech were then grouped into thirteen groups of similar POS. Each group was then assigned a single character that would be used in place of any POS within that group. POS defined by NLTK, as well as the groups created for this project, are defined in Appendix A. The goal is to utilize a string of the characters defined for each group in order to measure distance.

The initial data at this step is a list of sentences that were identified utilizing the process explained in Section 3.2. Each sentence is then run through NLTK's word tokenizer and POS tagger, to get a list of words and their associated POS. Then, for each word the POS is checked against the POS groups we've defined, and a character is appended to a string depending on what group the word falls under. Each sentence is therefore transformed into a string of characters, where each individual character signifies the POS that each word in the sentence was a part of.

3.3.2 Distance Formula Definition

Since each sentence has been reduced to a condensed string of characters, we can utilize the strings in order to define a sense of distance. Specifically, we utilize the Levenshtein Distance, or Editing Distance, as our metric for how similar/different sentences are from one another. This distance calculation involves calculating how difficult it is to transform one string into another. Character addition, deletion, and substitution are given particular weights and the lowest possible cost (based on weights) for the string transformation is calculated and stored. For the work presented in this paper insertion and deletion were both given a weight of 1, and substitution was given a weight of 2. No specialized substitution list was specified for this work (this an area for future work).

3.3.3 Distance Calculation and Clustering Creation

Once every sentence was transformed into a string of characters as discussed in Section 3.3.1, they are all compared to every other sentence using our distance function. The output of this computation is an NxN matrix of distance values, where each index refers to which sentence was compared to which other sentence. This matrix can then be run through Scipy's Linkage package to

accomplish the desired hierarchical clustering. A single link clustering was created from the distance matrix. The clustering was then visualized with a dendrogram.

4. Results

4.1: Matched Sentence Result.

There are three types of words in the final protein name list:

Protein name: The words that 100 percent matched the protein name with the database.

Related word: Words are not protein names but helpful or contain information that will help to identify protein names.

Unrelated word: words are not protein names and do not provide any information to track the protein name.

		immune sentinel proc such as type I and III antimicrobial protein
inflammatory	Unrelated word	The mechanisms that to microbial stimulat fully elucidated.

Table 1: Protein Extraction Example Results

4.2: Clustering Results

The dendrogram created following the steps discussed in Section 3.3 is shown in Figure 1. The sentences corresponding to some of the indices labeled in the leaf nodes of the dendrogram are provided in Table 2 and Table 3.

Word	Type	Sentence
TNF	Protein name	TLR stimulation of cord blood leukocytes results in a lower production of proinflammatory, Th1-associated cytokines (IL-12p70, TNF-α, IFN-α)
III	Related word	It is in close communication with AM and acts an

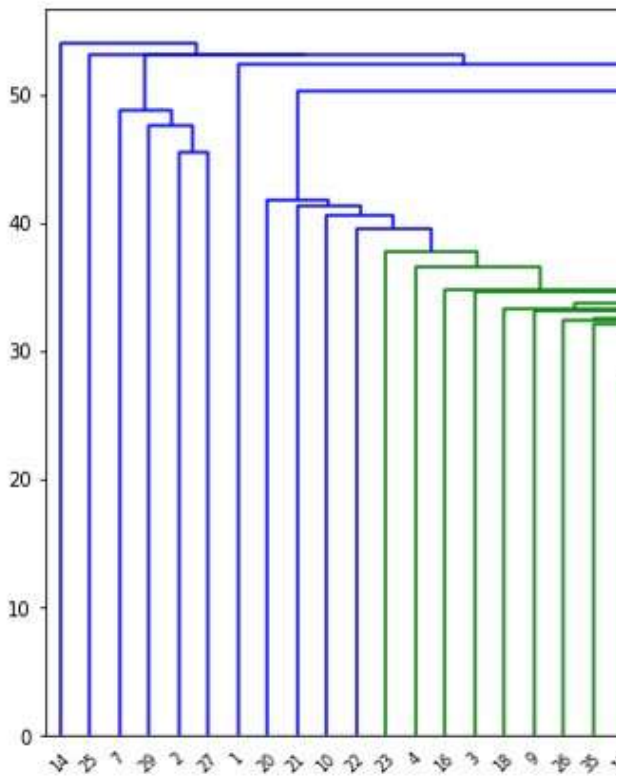

Figure 1: Dendrogram of Hierarchical Clustering done on Sentences.

2	27
'nninnvnidaninannnncanincdancnxvninnnianvrnnnrrr'	'idannnidannnidaanincaninnvidnnxinnnaannvvrr'

Table 2: Sentences identified by 2 and 27 in Figure XX

13	33	34
'anmviandncvn'	'nvdxanidaninn'	'idnnpvanennvnn'

Table 3: Sentences identified by 13, 33, and 34 in Figure XX

The sub-cluster defined by sentences 2 and 27 contained some discernible patterns. This includes the use of repeated “n”s, repeated “r”s, etc. There are also some pattern matches with things like “dan” and “nan”. The sub-cluster containing 13, 33, and 34 contained a bit more ambiguity in patterns or similarity. There is a slight repeat of “an” and “vn”, but not much else. One thing that Table XX and Table XX show is the clustering algorithms seemed to favor sentences with similar lengths when trying to pair them together. This is somewhat surprising considering the cost to insert and delete characters was defined as lower than the cost to substitute characters for each other.

5.Conclusion

The foundation of the project was based on the work of Kuo and Lin. They proposed seven types of regular expressions

to extract protein names. Then, we connected their work with the features of the name entity recognition and clustering developed clustering analysis for the sentence that contains protein name.

In the first part of the process of extraction, we located the Uniprot database as our main source to check the authenticity of protein names. Also, we used the paper "Innate immunity to respiratory infection in early life" as a reference for word extraction. The main techniques in the first part were to set up regular expressions and for loops. After checking the words with the database, we manage to get the words that contain protein names. Then, we used them to extract the sentences that contain protein names.

In the second part of the work, we used the concept part of speech to tag the sentences from the first part. After implementing the POS, all the sentences were transferred into a string of characters. Therefore, we were able to calculate the edit distance for each sentence. Combine the result of POS and edit distance, we built a matrix of distance values. Finally, we utilized the matrix with Scipy's Linkage package to set up hierarchical clustering.

References

[1] http://www.nhinbre.org/

[2] Huang-Cheng Kuo, Ken-I Lin, “Extracting Protein Names from Biological Literature,” *ACSIJ Advances in Computer Science: an International Journal,* vol 3, Issue 2, No.8, March 2014.

[3] https://www.statisticssolutions.com/directory-of-statistical-analyses-cluster-analysis/

[4] Lambert, Laura, and Fiona J. Culley. "Innate immunity to respiratory infection in early life." Frontiers in immunology 8 (2017): 1570.

[5] https://www.uniprot.org/

[6] https://stackoverflow.com/questions/7961363/removing-duplicates-in-lists

[7] https://www.nltk.org/book/ch02.html

Appendix A: NLTK POS tags and Group Creation

NLTK Defined POS:

- CC
- CD
- DT
- EX
- WDT
- FW
- IN
- JJ
- JJR
- JJS
- MD
- NN
- NNS
- NNP
- NNPS
- PDT
- POS
- PRP
- PRP$
- WP
- WP$
- RB
- RBS
- RBR
- WRB
- VB
- VBD
- VBG
- VBN
- VBP
- VBZ

POS Defined Groups for Cluster Analysis Data Creation:

Included POS Tags	Assigned C
‘CC’	‘c’
‘CD’	‘r’
'DT', 'EX', 'WDT'	‘d’
'FW'	‘f’
'IN'	‘i’
'JJ', 'JJR', 'JJS'	‘a’
'MD'	‘m’
'NN', 'NNS', 'NNP', 'NNPS'	‘n’
'PDT'	‘t’
'POS'	‘s’
'PRP', 'PRP$', 'WP', 'WP$'	‘p’
'RB', 'RBS', 'RBR', 'WRB'	‘x’
'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ'	‘v’