Automatic Protein Function Annotation Through Text Mining

Thesis by

Sumyyah Toonsi

In Partial Fulfillment of the Requirements

For the Degree of

Masters of Science

King Abdullah University of Science and Technology

Thuwal, Kingdom of Saudi Arabia

August, 2019

EXAMINATION COMMITTEE PAGE

The thesis of Sumyyah Toonsi is approved by the examination committee

Committee Chairperson: Robert Hoehndorf
Committee Members: Vladimir Bajic, Mikhael Moshkov

ABSTRACT

Automatic Protein Function Annotation Through Text Mining
Sumyyah Toonsi

The knowledge of a protein's function is essential to many studies in molecular biology, genetic experiments and protein-protein interactions. The Gene Ontology (GO) captures gene products' functions in classes and establishes relationship between them. Manually annotating proteins with GO functions from the bio-medical literature is a tedious process which calls for automation. We develop a novel, dictionary-based method to annotate proteins with functions from text. We extract text-based features from words matched against a dictionary of GO. Since classes are included upon any word match with their class description, the number of negative samples outnumbers the positive ones. To mitigate this imbalance, we apply strict rules before weakly labeling the dataset according to the curated annotations. Furthermore, we discard samples of low statistical evidence and train a logistic regression classifier. The results of a 5-fold cross-validation show a high precision of 91% and 96% accuracy in the best performing fold. The worst fold showed a precision of 80% and an accuracy of 95%. We conclude by explaining how this method can be used for similar annotation problems.

ACKNOWLEDGEMENTS

First and foremost, I thank Allah for granting me the luxury of education, health, family and friends. I then extend my appreciation to my mother and father who have wrapped me with care and warmth throughout this journey. Words cannot describe how thankful I am to my grandmothers who have prayed for me, cared for me and made me exquisite food. I am also thankful to my aunts and siblings, especially Sarah who has supported me in unimaginable ways. Thanks to the continuous encouragement provided by my friends: Abeer, Amani, Azza, Ghofran, Lama, Marwa, Mona, Nojoud, Roa, Sara, Sarah, Sakhaa and Wafaa, I was able to enjoy my time at KAUST and get through the difficult days.

On an academic note, I thank my advisor Dr. Robert Hoehndorf for his understanding and support in my thesis and for being accepting and enduring my shortcomings. I hold a great amount of gratitude towards Prof. Vladimir Bajic who has taught me so much in so little time and showed me how fun research can be if you do it for the right purpose. I also thank Prof. Mikhael Moshkov first for agreeing to be an enriching part of this thesis, but most importantly because he has taught me invaluable lessons and diversified my academic experience. Although I only took one course with Dr. Basem Shihada, his encouragement and honest advice were a constant source of support and for that, I am greatly thankful. Finally, I thank Prof. Mohamed-Slim Alouini who, as the vice dean, took the time to listen to my academic concerns and promptly helped me resolve them.

TABLE OF CONTENTS

Examination Committee Page **2**

Copyright **3**

Abstract **4**

Acknowledgements **5**

List of Figures **8**

List of Tables **9**

1 Introduction **10**

2 Related Work **13**

2.1 Computational methods 13

2.2 Text-based methods 14

2.2.1 Statistical properties of key terms 14

2.2.2 Dictionary-based features 14

2.3 Hybrid methods 15

3 Materials and Methods **16**

3.1 Data Retrieval 16

3.2 Dictionary-based feature extraction 17

3.2.1 Dictionary creation 17

3.2.2 Feature extraction 18

3.3 Dataset Creation 21

4 Results and Discussion **22**

4.1 Discarding illegible candidates 22

4.2 Training a logistic regression classifier 23

4.2.1 Modified stratified sampling 23

4.2.2 Training results 24

5 Concluding Remarks **28**

References **31**

LIST OF FIGURES

3.1 Extraction of dataset paragraphs 17
3.2 Creation of the weakly labeled dataset 21

4.1 Frequency of classes associated with words 22
4.2 Feature values of the negative and positive samples 23
4.3 Feature values of the negative and positive samples after discarding classes of low scores 25

LIST OF TABLES

3.1 Enumeration of the seven features used for classification 19
3.2 Example of a class's name and synonyms 20

4.1 Evaluation results 24
4.2 Evaluation results after filtering 25

Chapter 1

Introduction

Although experts have studied and complied many domain-specific knowledgebases and ontologies, the problem of instantiation remains unsolved. This is the case for the problem of annotating gene products with functions from the Gene Ontology (GO). Despite GO being available since 1998 and having received 6.4 million annotations of gene products to 4.4 thousand organisms, new annotations are discovered and recorded on a regular basis [1][2]. Hence, albeit the existence of information about gene product functions and their relations, knowledge about which gene products carry out these functions is incomplete.

Genes go through complex processes of transcription and translation to produce protein molecules that perform functions within cells [3]. As such, the annotation processe assigns functions to proteins rather than genes. The UniProt Knowledgebase (UniProtKB) maintains protein information of which are protein sequences and functions. But UniProt's rich content of more than 160 million protein sequences is mostly not annotated as less than 150 thousand proteins are annotated with function annotations [4]. Yet the knowledge of a protein's function is important as it finds many applications in molecular biology, genetic experiments, and protein-protein interactions [5].

Many studies were conducted to develop methods that solve the problem of annotating proteins with function. But as these methods are unreliable, all of the validated annotations were processed by expert biocurators [4]. Since many published bio-medical articles already contain experimental information that associate

proteins with functions, they serve as a rich and reliable resource for expert annotaters. However, the process of manual article-based annotation is tedious and unscalable. Therefore, findings of these articles are underutilized or overlooked because they are in free text form and are not directly available to researches.

Interestingly, the usefulness of validated annotations is not restricted to direct usage in their specific domains but exceed that to be a promising starting point for creation of automatic annotation tools. Many studies have utilized curated sets to build automatic annotators, but the results are far from perfect. To annotate protein functions, some researchers have used these datasets to validate methods that predict protein-function association based on protein sequences and structures [6]. On the other hand, different researchers focused on utilizing the abundance of information in the bio-medical literature to extract such associations from text with the help of these curated annotations [7]. In this thesis, we tackle the problem of automatic text-based annotation of GO functions to develop a more reliable method.

GO captures protein functions and divides them into three main categories: Biological process, molecular function and cellular component. Each sub-ontology has its classes which describe functions that maintain a hierarchical structure. The resulting graph of the classes and their relations is not a tree [2]. Each class in the ontology is characterized by its name, definition and synonyms. The annotation process follows these information and assigns a class to a protein only if it aligns with its information [8]. Hence, it is in our best interest to come up with a method that can mimic the behavior of an annotator who decides that a piece of text around a protein actually describes a particular class description of a function. As the complexity of assigning functions to different proteins which are mentioned in the same window is high, this research will be limited to annotating windows of text that have a single protein mention.

To achieve our goal of automatically annotating single protein mentions with

functions, we develop a dictionary-based method that first pools all candidates classes within a certain window then discards weak ones. Evaluating candidates is carried out by a logistic regression classifier that we train on a weakly labeled dataset. The main contributions of this thesis are:

- We develop a novel approach to annotate protein functions from text
- We leverage existing knowledge about protein functions that is found in GO to automate protein function annotation
- We evaluate our model by means of accuracy, specificity and sensitivity
- We show how this method can be used on similar annotation problems

Chapter 2

Related Work

The problem of associating proteins with functions has been approached by different methods. These methods can be distinguished by their main source of information: Textual sources, non textual sources and a combination of both. As such, these methods can be divided into three main categories: computational methods, text-based methods and methods that use a hybrid of the two.

2.1 Computational methods

As a protein's sequence dictates its function and structure, there have been studies that predict protein functions based on protein sequences. By observing protein variants and applying computational bioinformatic methods, researchers attempted to map protein sequences to functions [6]. On the other hand, different methods relied on protein structures to predict their functions [9]. Other studies leveraged protein to protein interactions to predict protein function with the observation that proteins that interact have similar functions [5]. Overall, these methods cannot provide strong evidence behind a prediction because of the lack of experimental evidence and hence remain unreliable.

2.2 Text-based methods

Text-based methods approach the problem in two different phases. The first being identifying windows of text that have protein function mentions and the second is identifying the protein functions in question. Many studies focus on a single aspect assuming the existence of the other. Most of the text-based methods are either dictionary-based or use features extracted via statistical properties of key terms.

2.2.1 Statistical properties of key terms

Studies falling under this category extract key terms from text based on their statistical significance by measures such as z-score and co-occurrence of terms with proteins. These terms then undergo natural language processing to further refine them according to their part of speech [10]. Alternatively, others use these terms to form vectors that are the concatenation of the individual statistical scores of these terms [7]. These methods incur the risk of ignoring indicative terms which happen to have a low statistical score. Researches applying these methods in isolation show no promising performance and have been outperformed by other methods.

2.2.2 Dictionary-based features

These methods use provided ontologies as dictionaries and use fast look-up algorithms to map given text to concepts, ConceptMapper and cTakes are examples of such methods [11] [12]. Many tools already provide such algorithms and they are usable on different ontologies. But as noted in a survey in 2014, these methods need to be further tailored to fit specific ontology characteristics and have only been tested on the names and synonyms of the functions [13]. Although differences exist between such systems, the underlying usage of a dictionary based on Ontology classes is persistent.

2.3 Hybrid methods

Hybrid methods use a combination of text-based methods and other computational methods. One such method relies on protein to protein interactions and uses evidence from the literature to improve precision [14]. Another method mainly relies on protein sequences and further uses text evidence in a consensus approach [15]. Hybrid methods leverage existing literature and additional information to associate proteins with functions. However, these studies mainly diverge their focus from the literature and only use it as means of validation if applicable.

Chapter 3

Materials and Methods

3.1 Data Retrieval

We use the Gene Ontology Annotation (GOA) dataset that is managed by the European Bioinformatics Institute [8]. GOA provide gene function annotation from GO to proteins in UniProtKB [4]. We retrieve annotations of human and mouse proteins as they are well-studied organisms. Each GOA entry annotates a single protein with a single function supported by an evidence code which gives information about how an annotation was made. There are six main categories of evidence codes: Experimental evidence, phylogenetic evidence, computational evidence, author statements, curatorial statements and automatically generated annotations. We focus on entries supported by experimental evidence as these entries provide the PubMed Central (PMC) ID of the articles used for annotation. A PMC ID is a standard identifier that uniquely identifies articles within the PubMed central that archives free access full-text scientific articles in the bio medical and life sciences fields [16].

Whatizit is a tool that enables users to look up mentions of entities in text by providing a dictionary of the desired entities and their mapping IDs [17]. We use Whatizit to perform entity recognition of proteins mentioned in the articles referenced by GOA entries that are supported by experimental evidence. After identifying protein mentions in articles, we extract the text surrounding these mentions. As individual sentences where proteins were mentioned provide limited information, we extract paragraphs as the description of a function can expand to multiple sentences. Since

we are not sure which paragraph in particular describes a certain annotated function, the dataset that we create is later weakly labeled.

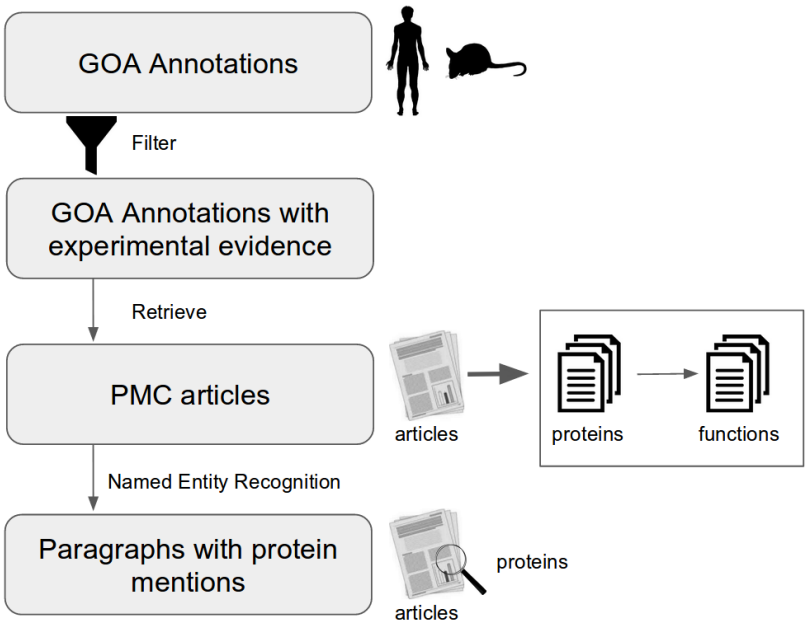

Figure 3.1: Extraction of dataset paragraphs

3.2 Dictionary-based feature extraction

Our approach uses a dictionary of GO to automatically annotate provided text around a protein with appropriate functions.

3.2.1 Dictionary creation

We create the dictionary in a novel way that allows us to pool possible function candidates around a protein based on the provided text. The dictionary does not include all classes of functions as some classes are too general or not intended for annotation purposes. To determine which classes to exclude, we base our decision on the information content score of each GO class that is provided by the SML toolkit [18]. The information content score ICC of a class c is defined as follows:

$$ICC(c) = -\log(Prob(c)) \tag{3.1}$$

Where:

$$Prob(c) = \frac{Number\_of\_annotations\_of\_class\_c}{Total\_number\_of\_annotations}$$

Hence, the lowest information content score would be of classes who have high numbers of annotations. Since an annotation of a child class is propagated to its parent classes, parent classes have higher probabilities and consequently lower information content score. This behavior is desirable as classes like biological process and molecular function cannot possibly be valid as immediate annotation.

After discarding the aforementioned classes, the words of each class's name, definition and synonyms are space tokenized. We remove stop words first then take each word and generate all of its forms of noun, verb, adjective and adverb as well as singular and plural forms if applicable [19]. After lower-casing the words, we use them to create a dictionary whose keys are these words and values are the classes where these words occur. For instance, the dictionary gives three classes of the IDs: GO:0007475, GO:0051124 and GO:0071340 for keys of *apposition*, *appose* and *apposed*. By including all forms of a word, we attempt to maximize the hit rate first and filter the results later.

3.2.2 Feature extraction

We use seven features to represent each candidate class and perform classification based on these features. The seven features gathered for each candidate class are summarized in the table below:

Table 3.1: Enumeration of the seven features used for classification

Number	Feature
1	The number of hit words of a class
2	The cumulative information content score of hit words
3	The number of words matched with class's highest ranking name\synonym
4	The fraction that the words in {3} make of the class's name\synonym
5	The length of the provided text
6	The number of hit words in total
7	The cumulative TF-IDF score of hit words

The first feature is the number of hit words of a certain class that is; how many words of the words that constitute this class's name, definition and synonyms have also appeared in the provided text. The second feature is the sum of the information content scores of a class's hit words. An information content score ICW of a word w is defined as follows:

$$ICW(w) = -\log(Prob(w)) \tag{3.2}$$

Where:

$$Prob(w) = \frac{Number\_of\_classes\_where\_w\_occurs}{Total\_number\_of\_classes}$$

The usage of the information content score in such manner enables us to measure a word's abundance in the ontology and hence gain an insight of how important it is. The third and fourth features are extracted by ranking the name of the class and its synonyms then choosing the one that is most complete among them. Hence, the third feature gives the number of matched words and the fourth feature represents the fraction that these words make of the highest ranked name or synonym. For instance, if the words ***apoptotic*** and ***clearance*** matched with a class that has the following name and synonyms:

Table 3.2: Example of a class's name and synonyms

Name	Apoptotic cell clearance
Synonym 1	Programmed cell clearance
Synonym 2	Apoptotic cell removal
Synonym 3	Efferocytosis

Then the class's name would rank the highest and the third feature would be equal to 2 as two words matched. The fourth feature would be 2/3 as the matched words constitute two thirds of the name.

The fifth feature describes the length of the provided text and the sixth feature is set to the number of hit words in general across all matched classes. Finally, the seventh feature describes the cumulative Term Frequency Inverse Document Frequency (TF-IDF) score of hit words for a class. The TF-IDF score of a word w in a certain class j is calculated as follows:

$$TF - IDF(w, j) = TF(w, j) \times \log\left(\frac{N}{DF(w)}\right) \tag{3.3}$$

Where:

$TF(w, j) =$ Frequency of occurrence of the word w in j

$N =$ Number of total classes

$DF(w) =$ Number of classes in which w occurs

This score gives us information about how important a word is to a class among all other classes. The score increases as the word is mentioned more frequently in a class but is penalized once it is mentioned in other classes.

3.3 Dataset Creation

To create the dataset, we use the paragraphs extracted as described in 3.1 and tokenize them after lower-casing and removing the stop words. We then hash the words of each paragraph to pool candidate function classes from the dictionary. That is, we check for intersections between a paragraph's words and the ontology classes' words. We then calculate the scores described in 3.2.2 to retrieve a feature vector for each candidate class. We weakly label the resulting vectors by the GOA annotation. As such, if a class appears as a candidate for a paragraph that has a mention of a protein that is annotated by this class in GOA with evidence referencing the source article, it is treated as a positive sample. Otherwise, all candidates other than immediate parents of positive classes were treated as negative samples.

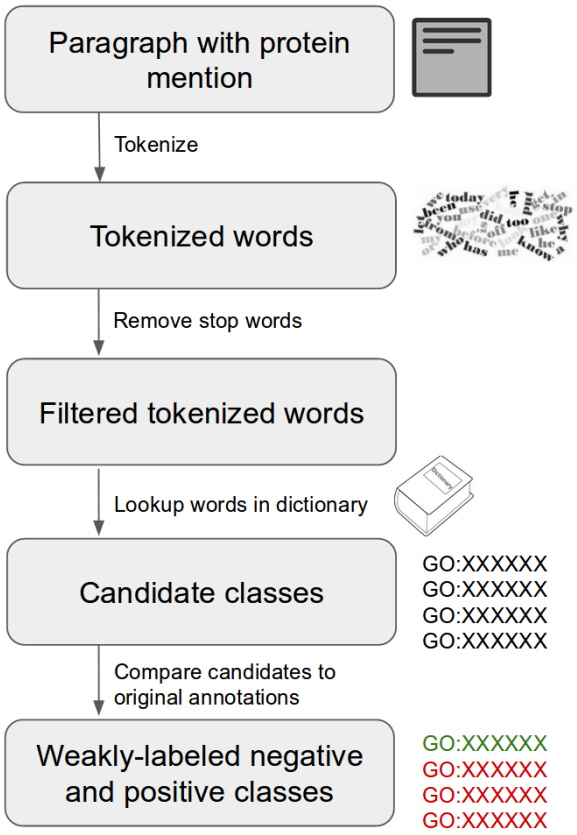

Figure 3.2: Creation of the weakly labeled dataset

Chapter 4

Results and Discussion

4.1 Discarding illegible candidates

The weakly labeled dataset we created in 3.3 is highly imbalanced as there are 150x times more negative samples than positive ones. A contributing factor to this imbalance is the high hit rate that pools many candidates just by one word that matches from their class.

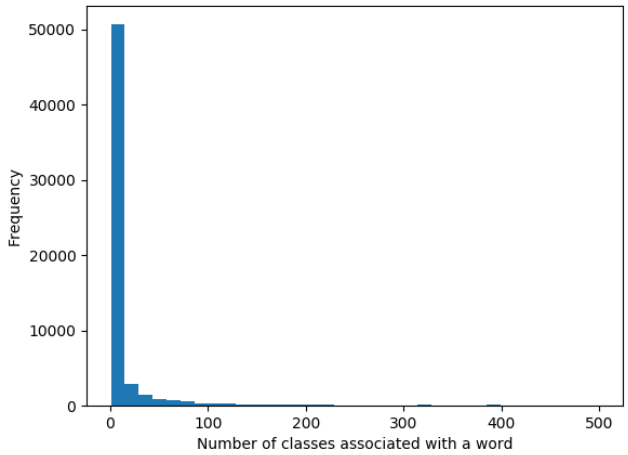

Figure 4.1: Frequency of classes associated with words

As the figure 4.1 demonstrates, there are words that occur frequently in classes and hence have a high number of associated classes. There are a few words that were not included in the figure that have more than a thousand associated classes. The imbalance is also affected by the matching of unrelated words that either add negative classes as candidates or raises the score of negative classes causing an imbalance.

To filter such negative classes, we discard classes that matched on a single word whose information content score is low. We further discard candidates whose average information content score is below a certain range. We found that discarding matches whose average information content score is lower than 1.5 gives the best behavior.

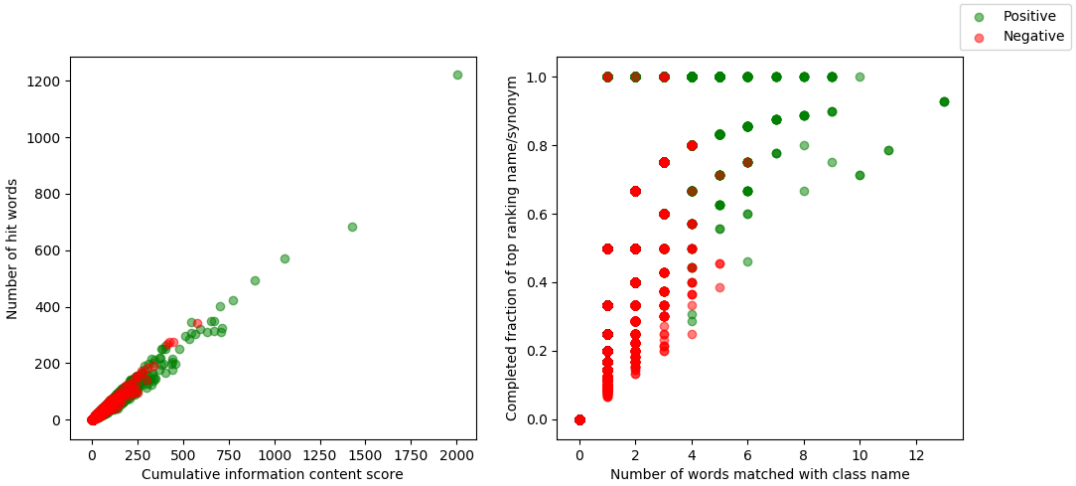

Figure 4.2: Feature values of the negative and positive samples

4.2 Training a logistic regression classifier

4.2.1 Modified stratified sampling

To balance the set before training the classifier, we perform a modified version of stratified sampling. We order the negative samples in an ascending order and choose samples of ranging values for the first four features. Since we have 6 thousand positive samples, we divide the negative set into six parts after ordering. We further divide each part into six and repeat this process four times in total. This results in 1,296 subsets of which we choose four random samples from each. This process results in 5,184 negative samples from the original negative set, we further choose random samples to total 6 thousand negative samples for training.

4.2.2 Training results

We use all the positive samples along with the samples mentioned in 4.2.1 to train and test a logistic regression classifier. We use Scikit's implementation of logistic regression to train our classifier [20]. To assess the model's performance, we use three types of evaluation measures: Accuracy, precision and recall. It is important to note that the values of the three measures reflect the performance of the model on the dataset used for training and testing. We evaluate our model using 5-fold cross-validation, the table below summarizes the results:

Table 4.1: Evaluation results

Fold	**Accuracy**	**Recall**	**Precision**
1	67%	66%	67%
2	75%	75%	76%
3	79%	78%	79%
4	75%	74%	75%
5	60%	58%	60%

The results show a low precision of 60% in the worst performing fold. After examining some instances, we realized that many of the positive instances have low scores. As such, it was difficult for the model to distinguish them from negative instances. To reach better worst-case precision, we apply strict rules for positive classes. We discard classes if they have less than 40% top name or synonym match. Furthermore, if a class's ratio between the number of hit words and the cumulative information content score is less than 2, we exclude it as well.

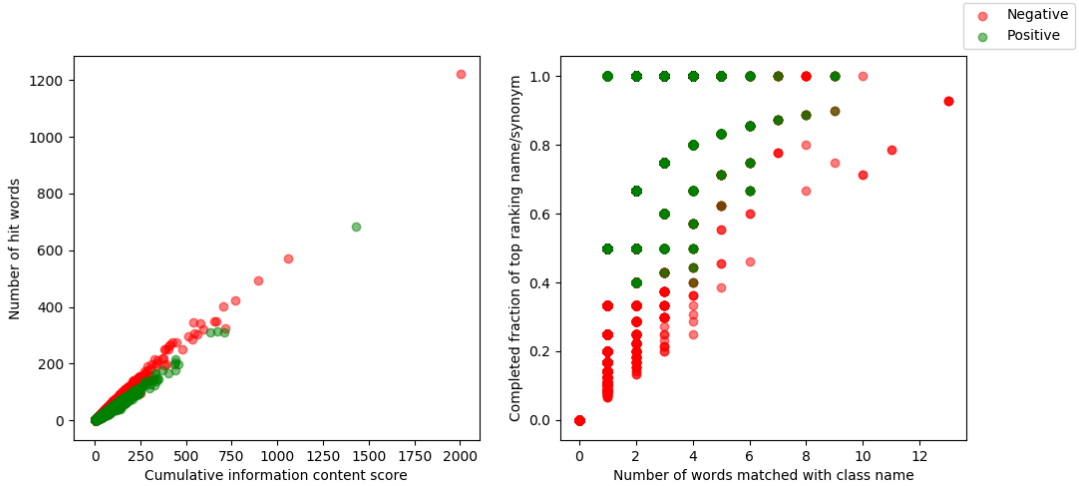

Figure 4.3: Feature values of the negative and positive samples after discarding classes of low scores

As figure 4.3 shows; the positive samples are more distinguishable from the negative samples than they were before the discarding process (see figure 4.2). This process translates the assumption that positive instances should have a considerable amount of resemblance to the annotated classes. It also ensures that positive instances are backed by a decent statistical evidence. This process drops the number of positive classes from 6 thousand to 1 thousand. We train a classifier to predict more precisely and the following table summarizes the results:

Table 4.2: Evaluation results after filtering

Fold	**Accuracy**	**Recall**	**Precision**
1	95%	71%	81%
2	95%	68%	87%
3	96%	73%	91%
4	94%	65%	82%
5	95%	74%	80%

Through restricting positive classes, we reach a higher precision. However, by applying restrictive rules on the created dataset, we lost many positive instances. Such a behavior is desirable as it is in our best interest to maximize precision. Having a higher precision means that the instances classified as positive are very likely to

actually be true positives. This strict discarding behavior enhances the accuracy as well because it makes it easier to distinguish negative classes from positive ones even if it sacrifices some positive instances.

This annotation problem is inherently difficult as it entails understanding natural language which is well-known to be a hard problem. Subtle differences in expression of the same function can be very confusing even to human beings. As such, although many more annotations were made based on the analyzed articles, our method was only able to detect 30% of the total annotations backed by experimental evidence that are provided by GOA. Moreover, cleaning textual data is also a challenging task that sometimes results in overlooking important terms. For instance, some articles use non-ASCII characters for special purposes which we ignore while processing. Moreover, tokenization plays an important role as it distinguishes keywords that are looked up in the dictionary.

While manually examining the extracted paragraphs, we have come across instances which were impossible to identify by our method. The reasons behind that include but are not limited to:

- Function annotations based on paragraphs that use synonyms of the words describing the annotated GO class
- Function annotations based on paragraphs that explain the function in a finer or coarser granularity than the annotated GO class
- Function annotations based on paragraphs that were poorly processed by tokenization
- Function annotations based on paragraphs that have protein mentions which were misidentified by Whatizit

Furthermore, special types of paragraphs tend to be more problematic than others. These paragraphs sometimes have more than one protein mention or are very long

that they lead to many negative matches. Additional difficulties arise when positive classes can be extracted by our method but have low scores which might be caused by any of the reasons highlighted above or a combination. It is also important to note that our method overlooks relations between the words in the provided text. That is, we lose information that the sentence structure provides as well as context information. Although we are aware that our method is not sufficient to identify complex descriptions of protein functions, we know that it performs well in other types of description. As such, we accept that our method will not be able to extract all of the available annotations in a provided text. Yet because we train our classifier to only accept positive cases of high statistical evidence, we can be more confident about the results that it yields. Hence, discarding positive instances which are not tractable by our method enables us to be more confident about the instances that our model classifies as positive.

It is also important to stress the scalability of our method. Our use of the dictionary enables us to generate candidate classes in a linear time of the number of words in the provided text. As each word is looked up in the dictionary, its classes are pooled in constant time. To create the feature vectors of classes, we use an additional dictionary whose keys are classes' IDs and values are the feature vectors. We further choose the highest ranking name/synonym and discard some classes which have low scores. To achieve both tasks, we go through each candidate class once. Since the ontology has 45 thousand classes, these operations are upper bounded by 45 thousand which is a constant. Hence, the overall time complexity is linear.

Chapter 5

Concluding Remarks

Although new protein sequences are constantly being discovered, little is known about their function. UniProt's knowledgebase of proteins has more than 160 million protein sequences but less than 150 thousand proteins were annotated with functions [4]. Knowledge about a protein's functions is essential to many applications in molecular biology, genetic experiments, and protein-protein interactions [5]. Many experimentally validated results already exist in the literature that associate proteins with their functions. Despite the existence of methods that automate text-based annotation of protein functions, the results remain far from perfect.

As such, we tackle the problem of automated annotation of protein functions based on text using a novel approach. We leverage the GOA annotation set to retrieve the articles on which the annotations were based. After identifying proteins in these articles, we extract the paragraphs where a protein mention took place. Through tokenizing the composing words of a paragraph, we match classes from GO to the tokenized words. We further assign seven scores to each candidate class to distinguish classes supported with enough evidence from others. We use these classes to create a weakly labled dataset based on the GOA annotations.

We train a logistic regression classifier on a weakly labeled dataset and observe a precision of 60% in the worst performing fold and a precision of 79% in the best performing fold. To reach a higher precision, we discard classes based on their scores. After discarding low scoring candidates, the results of the training showed a 80% precision in the worst performing fold and 91% precision in the best performing case.

We note that although the scores show an increase in recall, the number of positive classes was dropped because of the discarding process. Hence, the presented recall is bench-marked by the positively labled classes in the used dataset.

Upon applying our dictionary-based method, more negative classes are generated than positive ones. That behavior arises because many classes match on single words or words of low statistical significance. We also highlight a few reasons behind missing more than 70% of the original GOA annotations. Some important words were missed because of the tokenization process and some classes were missed because the description of the function did not use the terms used in the annotated GO class.

Although this project was tailored to solve the problem of annotating proteins with functions, this framework can be used for similar annotation problems. For instance, to solve the problem of associating a disease with phenotypes, a dictionary can be made based on the Human Phenotype Ontology. PMC articles can be retrieved based on the available annotations. From these articles, paragraphs of disease mentions can be extracted and the rest of the method can be applied as is. This new way of approaching the problem is easily reusable for similar types of annotating problems for which ontologies exist.

There is plenty of room for improvement and refinement in our method. By adapting more flexible methods of tokenization, more words can be matched. Additionally, we plan to resolve non-ASCII characters as well by mapping symbols like β to the letter "b". It is also important to further clean our dictionary by discarding parts of the definition of classes that describe the class in finer or coarser granularity than its name. Such parts of the definition cause a class to be incorrectly matched in a text that describes a completely different class. To enhance the training procedure, we aim to train an Artifical Neural Network (ANN) as ANNs have proven to outperform classical machine learning paradigms in many fields [21].

We further plan to test our method on a thousand PMC articles. To verify the results, we will compare any positive annotations with GOA annotations. If a positive annotations is not found in GOA, we will further test by comparing the function with functions of proteins that interact with the annotated protein. If the functions turn out to be similar, we will consider the annotation to be correct. Moreover, we are interested to annotate paragraphs which contain more than a single protein mention, possibly by leveraging Part Of Speech (POS) tagging.

REFERENCES

[1] Ashburner et al, "Gene ontology: Tool for the unification of biology," *Nature Genetics*, vol. 25, no. 1, May 2000.

[2] The Gene Ontology Consortium, "The gene ontology resource: 20 years and still going strong," *Nucleic Acids Research*, vol. 47, January 2019.

[3] B. Alberts, A. Johnson, J. Lewis, and et al, *Molecular Biology of the Cell*. Garland Science, 2002.

[4] The UniProt Consortium, "Uniprot: a worldwide hub of protein knowledge," *Nucleic Acids Research*, vol. 47, January 2019.

[5] J. Hou, *New approaches of protein function prediction from protein interaction networks*. London, United Kingdom: Academic Press, 2017.

[6] D. Fowler, C. L. Araya, S. Fleishman, E. Kellogg, J. Stephany, D. Baker, and S. Fields, "High-resolution mapping of protein sequence-function relationships," *Nature Methods*, vol. 7, August 2010.

[7] A. Wong and H. Shalkay, "Protein function prediction using text-based features extracted from the biomedical literature: The cafa challenge," *BMC Bioinformatics*, vol. 14, no. S3, 2013.

[8] R. Huntley, T. Sawford, P. Mutowo-Meullenet, A. Shypitsyna, C. Bonilla, M. Martin, and C. ODonovan, "The GOA database: Gene Ontology annotation updates for 2015," *Nucleic Acids Research*, January 2015.

[9] E. W. Stawiski, A. E. Baucom, S. C. Lohr, and L. M. Gregoret, "Predicting protein function from structure: Unique structural features of proteases," *Proceedings of the National Academy of Sciences*, vol. 97, no. 8, pp. 3954–3958, 2000. [Online]. Available: https://www.pnas.org/content/97/8/3954

[10] S. Rice, G. Nenadic, and B. Stapley, "Mining protein function from text using term-based support vector machines," *BMC Bioinformatics*, vol. 6, 2005.

[11] "The ConceptMapper approach to named entity recognition," Proceedings of Seventh International Conference on Language Resources and Evaluation (LREC10), 2010.

[12] Savova et al, "Mayo clinical text analysis and knowledge extraction system (ctakes): Architecture, component evaluation and applications," *Journal of the American Medical Informatics Association*, vol. 17, no. 9, September 2010.

[13] C. Funk, W. Baumgartner, B. Garcia, C. Roeder, M. Bada, K. B. Cohen, L. E. Hunter, and K. Verspoor, "Large-scale biomedical concept recognition: An evaluation of current automatic annotators and their parameters," *BMC Bioinformatics*, vol. 15, no. 1, 2014.

[14] S. Jaeger, G. Sylvain, L. Ulf, and R.-S. Dietrich, "Integrating protein-protein interactions and text mining for protein function prediction," *BMC Bioinformatics*, vol. 9, July 2008.

[15] R. You, X. Huang, and S. Zhu, "Deeptext2go: Improving large-scale protein function prediction with deep semantic text representation," *Methods*, vol. 145, August 2018.

[16] "PubMed Central," https://www.ncbi.nlm.nih.gov/pmc/, accessed: 2019-08-11.

[17] D Rebholz-Schuhmann, M. Arregui, S. Gaudan, H. Kirsch, and A. Jimeno, "Text processing through web services: Calling whatizit," *Bioinformatics*, vol. 24, no. 2, 2008.

[18] S. Harispe, S. Ranwez, S. Janaqi, and J. Montmain, "The semantic measures library and toolkit: fast computation of semantic similarity and relatedness using biomedical ontologies," *Bioinformatics*, vol. 30, no. 5, pp. 740–742, 10 2013. [Online]. Available: https://doi.org/10.1093/bioinformatics/btt581

[19] D. Chakravorty, "word forms," https://github.com/gutfeeling/word_forms, 2017.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[21] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2017.