UNIVERSITY OF CALIFORNIA SAN DIEGO

Towards Extracting Protein-Compound Interactions from BioChemical Patents

A thesis submitted in partial satisfaction of the
requirements for the degree of Master of Science

in

Computer Science

by

Kaiser Stefan Pister

Committee in charge:

Professor Leon Bergen, Chair
Professor Ndapandula Nakashole
Professor Zhuowen Tu

2019

The Thesis of Kaiser Stefan Pister is approved and is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank and acknowledge Professor Leon Bergen for all his help, advice and support with this project.

I would also like to acknowledge Cody Kim for his help in parsing data.

Additionally, I would like to thank Dr. Mike Gilson in the pharmacology department for his project idea and information.

Finally, I would like to thank my good friend Jimmy Yan for countless nights of listening to my hair-brain thoughts and patiently explaining many of the ideas in this work to me.

ABSTRACT OF THE THESIS

Towards Extracting Protein-Compound Interactions from BioChemical Patents

by

Kaiser Stefan Pister

Master of Science in Computer Science

University of California San Diego, 2019

Professor Leon Bergen, Chair

We present in this work a protein entity tagging and normalization process focused on data extraction from biochemical patents. The project acts a single stage in the pipeline of general chemical interaction extraction. Novel to this work is the character embedded approach to mention identification and normalization. Additionally, this is the first work to use a siamese network and a prototypical network to augment protein database normalization. Our results show that character embeddings provide a reasonable approach to protein entity extraction achieving up to 6% better results than previous work, and that normalization tasks can be improved significantly with a learned embedded space.

Introduction

Biochemical patents are arduous to read and comprehend. They can require multiple domain experts to parse out important details such as which proteins bind to what compounds in what ways. With recent advances in information retrieval and semantic analysis from neural networks, this project aims to reduce the workload on researchers in the field by performing parts of these tasks automatically. In this work we focus on patent protein extraction and further divide the task into three distinct subsections: *protein entity recognition*, *protein mention normalization*, and *target protein selection*.

Protein entity recognition is a specific form of named entity recognition. Here we locate all mentions of proteins within a document. To solve this problem, we build an LSTM classifier which distinguishes protein from non-protein sequences.

Tagging proteins is only a piece of the puzzle. Protein mentions are often not consistently referenced, leading to difficulty in finding related database entries. Thus, *protein mention normalization* is the process of converting a protein mention into a standardized name. We propose the use of deep learning to build embedded spaces which cluster mentions based on similarity.

Patents often mention many proteins, but are only concerned with the effects of one or two specific target proteins. We define the *target protein selection* task as recognizing which normalized protein or proteins are of highest significance to the patent. We use a set of heuristics combined with confidence scores from recognition to find the most important proteins.

This work presents a single pipeline to combine protein recognition, normalization, and selection into a usable process for researchers to quickly analyze and extract information from

patents without needing to delve too deeply into the dense writing itself.

Chapter 1

Background

In this chapter we describe various approaches that have been used in the past, analyzing their effectiveness and describing ideas on which the rest of this work is based. We concede that this is by no means an exhaustive list of works in the area, as that list would be too large to fit into this short paper.

Mention Recognition

There are many different obstacles in the way of recognizing a protein in biochemical documents. Primarily, the density of information can be daunting. Patents are packed with jargon which can be easily confused for important biochemical information. Without years of study and domain experience it would be impossible for a human to pull out the most important entities. As a result, researchers look to approaches which quickly learn in ways that humans cannot. In pursuit of protein entity recognition many groups have used traditional algorithms to varying degrees of success [8] [13]. These approaches often consist of regular expression- and rule-based dictionary lookups. In recent years, these approaches have been pushed out of practice and out performed by new techniques in deep learning [23]. We leave the discussion of these traditional methods to normalization where they are still comparable in performance.

In pursuit of well performing cheminformatics processing, BioCreative hosts competitive research workshops on a regular basis [10]. These workshops focus on specific tasks, such as chemical entity recognition, relationship extraction, document triage, and more, prompting

researchers to apply modern techniques to solve problems in the field of chemistry and biology. In the most recent two workshops, all of the top performing models leveraged advances in neural networks [11] [12].

The majority of work in this problem space make use of long short term memory neural networks which have proven to be quite effective in solving natural language processing tasks due to their ability to processes sequential data, extracting information from surrounding context. Since it has been discussed in great detail elsewhere, and the internal structure of an LSTM cell does not play a key role in this work's innovation, we refer the reader to [26] for a complete description of the mathematics behind these models.

Using LSTM architectures, research groups have been able to achieve great performance on chemical entity recognition [17]. However, the general task of tagging chemical entities proves to be easier to learn than tagging specifically proteins. At BioCreative V.5, the best teams achieved F1-scores above 90% when classifying chemical entities, while the GPRO (gene protein related object) tagging scores lagged behind at 81% [11]. General chemical entities often follow a stricter naming scheme than proteins, resulting in stronger patterns and signals which a network can identify [29].

Many deep learning approaches to recognition look fairly similar. Previous works largely build their neural network architecture from a set of LSTM nodes and a feature set from some combination of word embeddings and document information (e.g. frequency count or location in sentence) [8] [14] [20] [23]. These approaches have outdone previous results, however they also suffer from a couple of different problems. Due to the limits of manual annotation, the input data on protein names and feature sets of frequency, location, etc. are not necessarily representative of the full dataset [14]. Word embeddings do not convey the same information about form through the network, losing information about casing and structure of the sequence, forcing these to be used as separate features, leading to larger parameter spaces and more complicated networks [16].

Less common, character embedded models are another approach. In most work, they

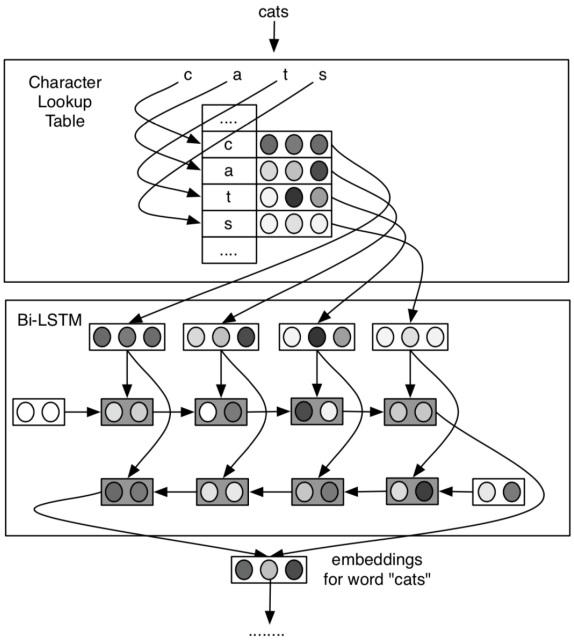

Figure 1.1. The structure of a character embedded bi directional LSTM neural network. Image from: [21]

manifest as a single feature of input data to be processed alongside a word embedding [23] [28]. We argue that the character model alone is enough to perform recognition, as has been shown in other application spaces [21]. In the work of Ling et al., they show that the form of a word can often convey enough information for classification tasks. These models rely on a non-arbitrary connection between the form and function of a sequence. The form of a character embedded model is shown in figure 1.2, which makes use of a character embedding flowing through a bi-directional LSTM network [21].

Some attempts at different architectures have been used in the past, a common thought is to use character embedded convolutional neural networks, however these approaches have not historically performed as well due to the importance of order in the sequence. In contrast, LSTM RNN architectures carry information directly through the order of the sequence [14].

The most common method of classification follows the BIEO system, tagging tokens as the Beginning, Internal, Ending, or Outside of a sequence [27]. Some approaches extend this

with a Singleton class as well to denote a single token sequence. Documents will be tokenized, often splitting into sentences, and then word sequences based on spaces, punctuation or special characters [13]. Depending on its location in a protein sequence, each token will then be tagged according to the BIEO scheme. Due to abnormal structure of chemical jargon, tokenization is not as simple a task as on traditional documents, preventing automatic solutions [5]. The result is that BIEO systems often have a smaller input dataset to train on, since the only reliable data will be hand curated. In trade for the smaller dataset, the BIEO system allows for precise learning of sequence structure that a binary system would not.

Mention Normalization

Due to its importance in database curation, named entity normalization is a heavily researched problem [22]. The most common approach has been dictionary look-up [5]. These normalization techniques range from a naive edit distance comparison between a target and the dictionary, and more complex regular expression comparisons. An alternative to dictionary look-up is rule based normalization, which describes a set of functions that transform the mention into a standardized form and then often use the standard form to perform dictionary look-up.

A list of common problems in normalization are shown in table 1.1 [27].

Perhaps the most commonly discussed disadvantage is that dictionary based approaches miss 100% of all new proteins. Without a constantly updating dictionary it is impossible to determine what a new protein normalizes too. Word embedding approaches have this issue as well, which we will discuss below. Stemming is a more difficult issue than commonly seen in linguistic tasks, as biochemical entities do not follow the same naming conventions as many normal stemming rules expect. The result is poorly stemmed names which cause more confusion than help. Acronym expansion is an issue in many different approaches, but can often be remedied by structural rules. For example, many documents will expand a full name of an entity before writing its acronym in parenthesis, allowing for easy detection. Orthographic variations account for a significant number of different representations of entities. Finally,

Table 1.1. Issues with normalization of protein names

Protein Mention	Standard Name	Issue
GHF1	GHF-1	Orthographic var.
GHF-1	Growth Hormone Factor 1	Acronym
Transcriptional Factor	Transcription Factor	Stemming
Antimicrobial Peptide	NDBP	Naming var.

naming conventions vary across regions and institutions, leading to tens of different ways to referring to the same protein. For a more in depth examination of normalization issues, we point the reader to Eltyeb et al. [5].

Beyond all of these problems, an even more pressing issue arises as noted by Leaman et al., which is that the process of name normalization often follows the imperfect process of entity recognition [16]. The pipelined nature of this task leads to cascading errors which can make any approach look worse than it would in an isolated case.

In 2010, BioCreative proposed a research track to normalize protein and gene names. The results of the track were dismal, with TAP-k scores of under 0.45 (k=20) [22] (an equivalent F1-score of 46.56% [27]). At the time, almost all approaches were constrained to either regular expression- or rule-based dictionary retrieval, with only a couple submissions making use of machine learning techniques such as conditional random fields, and support vector machines. More recently, there have been approaches to perform normalization with deep learning approaches [15][16][27]. These approaches use embedded spaces to transform proteins or protein representations into some new space, and then use a pairwise ranking function to find the nearest neighbor.

All of these works are using pre-trained word embeddings, sometimes with feature sets which include character representations. Since we build our features from only the character embeddings of sequences, there are other avenues of normalization available to our approach. In 2005, Chopra et al. proposed a new network architecture called a siamese network [1]. Figure **??** displays the structure of a siamese network. The purpose of this network is to build an

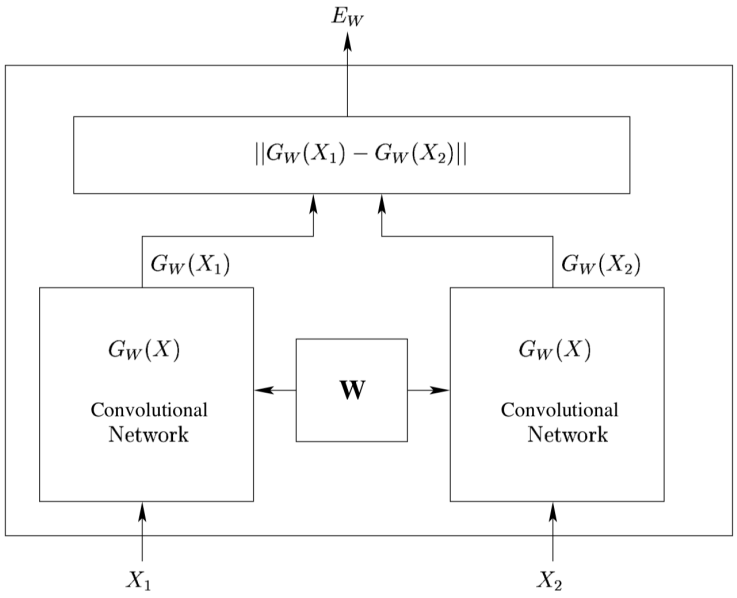

Figure 1.2. The structure of a siamese network with convolutional encoders. Image from: [1].

embedded space on the input data by embedding the data and optimizing loss on a distance metric between different classes of input data. The work was improved upon in [9]. In constructing this embedded space on character embeddings of normalized proteins with some of their mentions, we can find the nearest neighbor of an unknown embedded protein mention and treat that as the normalized name. This method acts in a very similar way to clustering techniques. For distance metrics, we use cosine similarity and Euclidean distance between points. The loss function is a contrastive loss function.

Hard negative mining (HNM) gives better results in similarity prediction as it furthers the distance between similar classes [7]. With HNM, the loss of each batch is penalized for the distance between each positive pair of input, and the similarity between each input and the closest non-identical element of the same batch. The result is that the network will build large inter-class distance, while keeping intra-class distance small.

More recently, Snell et al. proposed Prototypical networks for the task of few-shot or zero-shot classification [25] of handwritten characters. Prototypical networks aim to find the prototype of a class which acts as the center of a cluster by averaging the embeddings of each

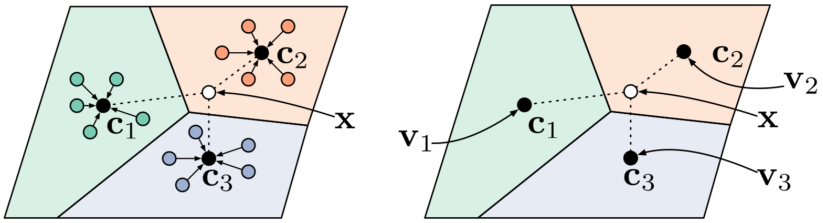

Figure 1.3. Predicting an input value X using few-shot and zero-shot prototypes. Image from: [25].

member in a class. Shown in figure 1.3, the prototypes of each class are used to predict on input. Then when embedding a new input with the network, their results show, the nearest prototype is a more accurate predictor of the class. Their structure also follows a contrastive loss function built on Euclidean distance. Building on their work, we apply a prototypical embedding network to the normalization process and compare the results in chapter 3.

Relation Extraction

This work acts a stepping stone towards solving the larger problem of extracting relationships between proteins and compounds in biochemical patents. We do not address the relationship extraction task, though there has been other recent research on this task. At the most recent BioCreative workshop, an entire track was dedicated to this problem [12]. Bert was introduced in 2018 by Devlin et al. as a new approach to solve many different natural language processing tasks. The key to its success is the idea of using a layer of transformers which perform significantly better than previous state of the art [4]. Following Bert, Lee et al. propose a domain specific BioBert which takes the same architecture but pre-trains with a different biology-tailored dataset [18]. Using different input data gives the Bert model a strong contextual understanding of biological terms which has given similarly impressive improvements to retrieval tasks in the biochemical domain. Some other recent applications use transfer learning with specialized word embeddings to extract relationships [3]. For more complete information, we refer the reader to [23].

Chapter 2

Methodology

In the following chapter, we describe the various approaches to solve each step of protein extraction. We conclude with some unsuccessful methods and the knowledge gained from them. As mentioned previously, we segment the task of protein extraction into three parts: *protein recognition*, *protein mention normalization*, and *target protein selection*.

Protein Mentions Recognition

To recognize protein mentions in documents, we build a character embedded bidirectional LSTM classifier. The model is inspired by Ling et. al's work and does not innovate on their structure. Our goal is to make use of character embedded models in the biochemical application space, since previously most work focuses on word embedded models.

Dataset

We use a dataset of protein names gathered from the UNIPROT KB gold star database [2]. This dataset includes information about each protein, from which we take the full name and alternative names for each entry. This dataset includes 64,496 unique proteins and approximately 240,000 total names. There are almost twice that many entries in the database, however there are duplicates for each species where the protein is found, which we do not include as the mentions are the same. The full set of protein names is treated as our positive dataset. To generate a set of negative sequences, we scrape bio-medical Wikipedia entries. We choose bio-medical pages to

have a better contextual match to the content in patent data, hopefully negating generic scientific terms while emphasizing proteins. With hundreds of pages as negative input, we have millions of negative data points. As an additional negative dataset, we generate a blended set of data which consists of fractions of protein names combined with parts of Wikipedia sequences. These blended samples are randomly generated and are meant to help the classifier find the precise location of a protein mention without including the surrounding words. We also include the patents themselves as a negative dataset. This will train the model to dismiss common sequences in the patents, while highlighting rarer sequences. Finally, we include a list of metabolites (small chemical compounds) as a negative dataset. Metabolites represent the most common chemical entities in our patent dataset, so we use them with a negative label to better target proteins.

With each dataset, we convert it into a set of uni-, bi-, and tri-gram sequences which are then fed into the model with their respective labels. We restrict our labels to positive or negative, ignoring the location of a token in a protein mention. We hypothesize this will improve the network's ability to reorder parts of proteins, however it comes at the cost of less precise boundaries of mentions.

We also perform casing and ordering transformations on the positive dataset. The case of a protein can convey distinctive information, but is not standardized across usage, thus we allow for all forms of casing to be positive (e.g. Adenosine A1, adenosine A1, adenosine a1, Adensosine a1). Additionally, we compute some permutations of the protein names to allow for different usage (e.g. adenosine a1, a1 adenosine). With these transformations, we capture many different styles of writing protein names.

Architecture and Parameters

The architecture of our model is a simple single layer bidirectional LSTM RNN. We use a one hot character embedding model to convert each sequence to a network readable input, then create a single hidden layer of 120 LSTM nodes which feed forward into a fully connected layer that outputs to 2 classes. We use a ReLU non-linearity function on the results. To convert the

LSTM layer to the fully connected layer, we average across all the hidden layer weights. Similar to other work, we use a learning rate of 0.001 and an ADAM optimizer. The loss function is a categorical cross-entropy loss. These parameters were chosen after performing a grid search across layer size, learning rate, and batch size.

Normalizing Mentions

Protein mentions are notoriously diverse. As mentioned previously, they can be represented with abbreviations, in different term order, with different spellings, or simply different colloquial names. Thus a single protein may be discussed with five, ten or even twenty different possible descriptions. When we identify a single mention, we cannot assume that the form of that mention is the normal form that will be stored in a dictionary. We solve this problem with three methods and evaluate their effectiveness in the next chapter.

Dataset

We utilize the same positive dataset as the recognition step, but modify its structure to distinguish classes of proteins. For each protein, we define the class name to be its Full Name according to UNIPROT, and list every other version of its name as an example of the class. With about 64,000 proteins, and 240,000 mentions, there are an average of 4 entries in each class, ranging from a low of 1 entry to a high of 23. All the protein names are transformed to lower case ASCII for simplification, which does lose some information on UTF-8 encoded characters. Using the Python package `unidecode`, we convert any UTF-8 characters to their closest ASCII approximation to minimize any differences. We divide our dataset two different ways. First we build a zero-shot dataset, in which our training data will see 60% of the classes, and every mention representing those classes. The other 40% is divided between validation and test sets. This dataset is zero shot because the model will not have seen any elements of these classes and therefore will have no prior on where to embed them. Then we build a few-shot dataset in which we divide each class into a 75-25 training-test split based on mentions. We discard any classes

that have fewer than 4 representations. This will train the model to embed based on parts of classes, and evaluate the model on how well it learned to embed unseen mentions of seen classes.

Levenshtein Distance

Our first normalization method calculates the sequence edit distance between two input strings using the Levenshtein distance metric [19]. This distance function attributes 1 point of error for each deletion, insertion and substitution.

Siamese Embedding

Our major contribution to this space is our work in training embedded spaces for normalization. As discussed in chapter 1, this is a widely studied field with many different application spaces. As far as we are aware, this is the first application of these deep learning embedded spaces to this problem. We again train a bidirectional LSTM RNN to encode protein mentions which refer to the same protein class into similar locations. Our architecture is very simple, we have a layer of 120 LSTM cells feeding into a fully connected layer which outputs into a 100 dimensional vector and then a ReLU function. The resulting vector is our embedding of the input. The siamese network's goal is to learn an embedding function such that two elements of the same class (i.e. two mentions of the same protein) will be a small distance apart, while two elements of different classes will be a large distance apart. The structure of our contrastive loss function resembles previous work [1], but we tailor it to our task, training on multiple distance functions as well as optimizing with hard negative mining techniques.

We train our model with both a Euclidean and a cosine similarity distance function and evaluate their different performances. Cosine similarity provides simpler and more efficient implementations, but can lead to poorer performances [25]. As is necessary for all cosine similarity distance functions, when training we add more more length normalization step to all embeddings.

Using hard negative mining, we target the poor performers of each batch to ensure that

the network learns a robust embedding. When using Euclidean distance, we adjust the hardest negative mining to be what we call harder negative mining. This is a stochastic approach in which we sample a subset of the batch, and only find the hardest element of that subset. While this will in practice potentially not create an optimal embedded space, the random nature of it does well enough, and it significantly improves performance of training.

With each positive and negative example we use a contrastive loss function to improve performance over time. Once we have an embedded space, we are able to find the nearest neighbor of an protein mention sequence by running it through the network. If the network has created a good embedding, then the mention will be close to the other proteins it relates to.

Prototypical Network Embedding

Another approach that we briefly explore to create an embedded space is following the idea of [25], in which we create a prototype representation for each protein. Then, similar to the siamese embedding, we can feed any sequence into the network and attempt to find the closest prototype. This prototype will correspond to the normalized entry in the database. We use the open source code accompanying the paper by Snell, accessible at `https://github.com/jakesnell/prototypical-networks`, only changing the encoder to follow suite with our other models, and adapting the dataloader to accept our dataset.

Target Protein Extraction

With the above architectures, we pass a patent document through a few preprocessing steps and the recognition model to receive a list of probable protein sequences. Using the DBSCAN clustering algorithm and a Euclidean distance fucntion over the embedded space of our Siamese network, we cluster the probable sequences into groups of similar mentions [6]. Then we weigh the clusters based on an average confidence score of its members and output the best two clusters. We give extra weight to protein mentions in the title, abstract, and reference sections as well as sections surrounding certain keywords which we identify as

often correlated with the target protein mention. Currently these sections surround the keywords "patent," "invention," "receptor," and "inhibitor."

Other Explored Areas

Throughout this process, we encountered and explored a variety of other approaches to solve the recognition problem. We find it important to document science in the presence of failures to allow for the better distribution of ideas and to hopefully provide ideas and guidance to others attempting to solve the same problem.

TF-IDF

Term Frequency Inverse Document Frequency can be a powerful metric in finding words that are important in a document. The metric compares how many times a term appears in a single document relative to the total number of times it appears in a corpora. Thus, a word that shows up 5 times in a document, and only 5 times in the entire corpora (notably the same 5 appearances in the document we are testing), will have a TF-IDF score of 1.0. Inversely, a term that is in a document 5 times, but appears 1000 times in the corpora (not uncommon for simple words), will have a TF-IDF of 0.005. This approximation should give a score of how important a term is to a given document. Here are some issues we found that occur with TF-IDF.

- The corpora isn't large enough. This is the most common issue. Without a large set of words to normalize out slightly uncommon terms, words like "office" end up with very high scores as they don't commonly appear in the text. In the same vein, this issue causes problems when documents are written by different people. We write very differently based on geolocation and education and will use very different sets of vocabularies even within the same language. We have 1515 patents, which is not enough to block out common words.

- The proteins are rarely similarly mentioned. As discussed in the normalization section, different groups might refer to the same protein in many ways, and even within the same

document a protein will be mentioned in different ways. This problem is more general to include the idea of non-stemmed words. An example is seeing the term ”word” and ”words” as different terms. There exist approaches to stem words, but this can often cause problems with protein names. Additionally stemming is not a correct normalization function for proteins, so that problem would persist regardless.

- The proteins are very infrequently mentioned. This might not actually be a problem if the protein isn't mentioned in any other documents, however proteins often share pieces of terms. ”Kinase” for example is a very common suffix term of a protein name. This leads to a very low score for any terms including ”Kinase” despite the definite importance. Patents are often focused on the new compounds being invented and aren't focused on the proteins, as a side effect the proteins aren't mentioned as much and might not be unique to each document.

TF-IDF does provide some interesting insights into the problem but ultimately is not the best approach. One thing we do consider for future work is an ensemble method including the TF-IDF score for each term, however the above problems would need to be addressed before that work would be conducted.

Convolutional Neural Networks

Another logical approach would be a CNN. Given we are already working with a character based LSTM model, it is not too far a step to experiment with a CNN. This follows the from the idea that our architectures work for any form of encoding method, however the results will find some encoders to be better than others. For this reason we do conduct some early experiments, however we found no significant differences from our LSTM model results, and continued efforts in developing the LSTM model instead. We do concede that this could be further explored, however as pointed out in [16], convolutional networks will suffer from lack of sequence ordering details.

Chapter 3

Results

We analyze our different models under the usual heuristics of precision, recall and f-measure in order to assess their correctness. Both the recognition and normalization models are evaluated individually. After we show a combined evaluation of how well they solve the protein extraction task on a dataset of patents.

Recognizing Mentions

We compare our results here against other works on the BioCreative V & V.5 protein tagging competition results. On the GPRO ChemDNER dataset, the best reported results were an F1-score of 81.2%, with a best precision of 80% and a best recall of 85% on different runs [13] [11]. These results were generated from various models ranging from CRF LSTMs to dictionary look-ups. Using the same evaluation metric, our recognition model under performs in comparison to state of the art. We are achieve an F1-score of 73.08% on the GPRO dataset.

Error Analysis

We find the strict nature of the evaluation to cause a significant number of errors in our model. The recognizer would, for example, tag the sequence "human p55 TNF" as a single protein sequence instead of two. This style of error, in which the model is off by a single word or contains two proteins in one, account for more than 70% of the false positives in our predictions.

Below we perform a more in depth analysis of our model, allowing for these errors to

Table 3.1. Results of Protein Entity Recognition

Evaluation Type	Precision	Recall	F1-Score
Our Model	85.10%	89.05%	87.03%
Best Prior	83.95%	78.66%	78.7%

be treated as correct mentions, resulting in a significantly higher F-score. To reiterate the type of error we allow, for the sentence "We use human p55 TNF as an inhibitor of compound Z," the desired tagging would be $proteins = \{\text{“}human\ p55\text{”}, \text{“}TNF\text{”}\}$.Our tagger might produce $proteins = \{\text{“}human\ p55\ TNF\text{”}, \text{“}p55\ TNF\text{”}, \text{“}TNF\text{”}\}$.With relaxed matching "p55 TNF" is not considered a false positive and "human p55 TNF" satisfies the "human p55" protein name. With trigram sequences, we will never allow a relaxed match which is more than a single token in distance away from truth. We could potentially reduce this problem by tagging with the BIOE system. The results of these two evaluations are shown in table 3.1.

The results of best prior are from the BioCreative V.5 which evaluated the performance of recognizing normalized and non-normalized protein and gene mentions in abstracts [24].

We evaluate our model by grabbing the matches which are above a 95% confidence score of being a protein to reduce the number of false positives, at the cost of slightly larger amount of false negatives. We make this choice because definitely having some proteins is much better than having to sift through a lot of bad data.

As shown in the table, with relaxed matching, we are able to significantly improve upon the results of previous work. Under strict matching, we have many more false positive mentions, as shown in the example above where "p55 TNF" will be considered an erroneous tag causing a significantly lower precision score. We perform a manual review of these false positives, finding 28/100 to be true false positives while 72/100 fall under the same situation of "p55 TNF" in that they represent parts of true protein names. We find it more important to tag an approximate location of a mention over the exact bounds of a protein so we accept the relaxed matching idea.

We acknowledge that other groups could potentially also improve their results when

using the kind evaluation metric.

Normalizing Proteins

There have been fewer attempts at normalizing protein mentions. As such, we compare against a couple other methods, and create a new baseline to evaluate our new models.

A New Baseline

The baseline we create is a binary test to check if two protein mentions refer to the same protein or are different proteins. Due to the binary nature of the test, a random guesser would perform an even 50%. We use a test dataset containing an even split of positive and negative pairs. For each test we are optimizing for the highest quantity of correct results.

A naive approach to solving this baseline is to compare the Levenshtein distance between the two input mentions against a threshold, returning true if the distance is small and false if it is large. This approach has the advantage of catching small differences between protein mentions from casing, reordering etc. It suffers on acronyms (since there is a large edit distance between A1A and A1 Adenosine) and completely different mentions. We perform baseline tests at a range of different threshold values, and also on a dataset without acronyms. With optimal parameters, the approach achieves an accuracy of 64.34%. We also evaluate this approach on the dataset without including any acronyms to better favor the metric, improving accuracy to 67.50%.

Table 3.2 shows common examples of errors when the Levenshtein distance metric fails to evaluate similarity. When choosing a threshold, there must be a balance between allowing enough distance between two similar proteins, while minimizing the false positives that will show up as in the third row of the table. Additionally, as seen in the second row, some protein mentions are sufficiently different that no linear distance metric would suffice.

Table 3.2. Examples of Levenshtein Distance Metric Errors

Type	Proteins	Reason	Distance
False Negative	nlrr-4 Leucine-rich repeat neuronal protein 4	Acronym	34
False Negative	NDBP-4.17 Antimicrobial peptide UyCT3	Different Name	27
False Positive	f18 PAP	Too short	3

Siamese Network Performance

The first model we propose to solve this problem is a zero-shot siamese network architecture. As described in chapter 2, we use the siamese network to create an embedded space. After embedding both input mentions, we compute the Euclidean distance between them and compare that distance against a threshold. Similar to Levenshtein distance, we return true if the distance is under the threshold and false otherwise. We range over many threshold values to find the optimal threshold of 0.6. At the selected threshold, the siamese network significantly outperforms the Levenshtein distance, achieving an accuracy of 79.05%.

The errors of the siamese network are less straightforward compared to the Levenshtein model as it is not clear the model the network learned for the problem. The performance of this network is evaluated on proteins for which it has seen none of the classes or mentions.

Notably these are zero-shot models. The network trains on an entirely separate dataset from the test data, meaning it has seen no forms of each test protein.

We additionally train a few-shot siamese network in which we give some examples of each protein class and mentions. Since each protein has many names, we define one name to be the standard form, then split the remaining names into training and test. In evaluation we test against the held out set of mentions which the network has not seen. Our best performance with this network is 87.43% accuracy. We will use this as a classifier in the future.

It is important to note that all tests are run on valid data. We do not attempt to normalize

Table 3.3. Normalization Task Results

Model	Accuracy
Levenshtein	64.34%
Levenshtein (no acronyms)	67.50%
Euclidean siamese (zero-shot)	**79.05%**
Euclidean siamese (few-shot)	**87.43%**
Cosine Sim. siamese (zero-shot)	76.12%
Prototypical (zero-shot)	63.89

erroneously tagged sequences. This is in an attempt to evaluate the normalization processes instead of the tagging process and not allow for cascading errors as talked about in [16].

Prototypical Network Performance

Finally we attempt to solve the zero-shot normalization task with a prototypical network. On the same task the network does very poorly, getting at best 63.89% accuracy. In future work, we would experiment more thoroughly on the parameter space of the network in attempts to improve this score.

Targeted Protein Results

In order to evaluate the practical results of these models, we use them to predict the target proteins of 200 patents. The patents were manually annotated with the set of import proteins mentioned in that patent. Once we tag, normalize and cluster the proteins in each document, we manually check these 200 patents for accuracy of the top clusters. We find that in 48% of the patents, the top rated cluster contains one of the target proteins, while 25% more were contained in the second highest weighted cluster. This level of extraction would still certainly require human guidance, however it provides a starting point that would potentially help expedite the processes for domain experts.

Chapter 4

Conclusion and Future Work

Our performance in both recognition and normalization show that character based models have a place in protein extraction. Using larger datasets than manually annotated abstracts gives us a wide scope of learning material from which our normalization model does quite well. The range of methods and applications covered in this project under the time constraints given lead to a cursory exploration of many parts in the biochemical natural language processing space. There are many parts of the project where we would look to improve or expand our work as described in the following list:

1. **Interaction Extraction** With strong chemical compound extraction techniques, our protein extraction techniques and the semantic understanding ability of BioBert, we would like to build a single pipeline for full relationship extraction. This would be a large task that would greatly benefit documentation, triage, and exploration of new proteins and protein relationships.

2. **Prototypical Networks** The classifier built on a prototypical network performs quite well on the protein set, however it could be improved on the baseline. There is more work to be done on fine tuning the parameter space as well as the encoding methodology of the network.

3. **Building a stronger pipeline** By no means do we believe any part of this work to be the final stage in the research of its field. Converting our pipeline into a modular work so that

we could easily incorporate improvements from other protein recognition, normalization or targeting steps would make the project more serviceable and usable by researchers.

4. **De-noising Protein Mentions** Using the same normalization techniques, we would like to training a network to remove noise in erroneously tagged protein mentions. This would help solve some of the protein recognition problems we encounter, converting “a TNF” to “TNF.”

5. **Morpheme embedded Networks** There has been recent research into training networks on the morphemes of words rather than the characters or full words themselves. We would like to take this approach to protein names as well, since the morphemes in a name often convey useful information.

Bibliography

[1] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546, 2005.

[2] UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic acids research*, 47(D1):D506–D515, 2018.

[3] P Corbett and J Boyle. Improving the learning of chemical-protein interactions from literature using transfer learning and specialized word embeddings. *Database*, 2018, 7 2018.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Safaa Eltyeb and Naomie Salim. Chemical named entities recognition: a review on approaches and applications. *Journal of cheminformatics*, 6(1):17, 2014.

[6] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[7] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. *arXiv preprint arXiv:1707.05612*, 2017.

[8] Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48, 7 2017.

[9] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[10] Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Alfonso Valencia. Overview of biocreative: critical assessment of information extraction for biology, 2005.

[11] Martin Krallinger, Martin Pérez-Pérez, Gael Pérez-Rodríguez, Aitor Blanco-Míguez, Florentino Fdez-Riverola, Salvador Capella-Gutierrez, Anália Lourenço, and Alfonso Valencia. The biocreative v. 5 evaluation workshop: tasks, organization, sessions and topics. 2017.

[12] Martin Krallinger, Obdulia Rabal, and Saber A Akhondi. Overview of the biocreative vi chemical-protein interaction track. In *Proceedings of the sixth BioCreative challenge evaluation workshop*, volume 1, pages 141–146, 2017.

[13] Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M. Lowe, Roger A. Sayle, Riza Theresa Batista-Navarro, Rafal Rak, Torsten Huber, Tim Rocktäschel, Sérgio Matos, David Campos, Buzhou Tang, Hua Xu, Tsendsuren Munkhdalai, Keun Ho Ryu, SV Ramanan, Senthil Nathan, Slavko Žitnik, Marko Bajec, Lutz Weber, Matthias Irmer, Saber A. Akhondi, Jan A. Kors, Shuo Xu, Xin An, Utpal Kumar Sikdar, Asif Ekbal, Masaharu Yoshioka, Thaer M. Dieb, Miji Choi, Karin Verspoor, Madian Khabsa, C. Lee Giles, Hongfang Liu, Komandur Elayavilli Ravikumar, Andre Lamurias, Francisco M. Couto, Hong-Jie Dai, Richard Tzong-Han Tsai, Caglar Ata, Tolga Can, Anabel Usié, Rui Alves, Isabel Segura-Bedmar, Paloma Martínez, Julen Oyarzabal, and Alfonso Valencia. The chemdner corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics*, 7(1):S2, 2015.

[14] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[15] Robert Leaman, Rezarta Islamaj Doğan, and Zhiyong Lu. Dnorm: disease name normalization with pairwise learning to rank. *Bioinformatics*, 29(22):2909–2917, 2013.

[16] Robert Leaman and Zhiyong Lu. Taggerone: joint named entity recognition and normalization with semi-markov models. *Bioinformatics*, 32(18):2839–2846, 2016.

[17] Robert Leaman, Chih-Hsuan Wei, and Zhiyong Lu. tmchem: a high performance approach for chemical named entity recognition and normalization. *Journal of cheminformatics*, 7(1):S3, 2015.

[18] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*, 2019.

[19] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, February 1966.

[20] Sangrak Lim and Jaewoo Kang. Chemical–gene relation extraction using recursive neural network. *Database*, 2018, 2018.

[21] Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.

[22] Zhiyong Lu, Hung-Yu Kao, Chih-Hsuan Wei, Minlie Huang, Jingchen Liu, Cheng-Ju Kuo, Chun-Nan Hsu, Richard Tzong-Han Tsai, Hong-Jie Dai, Naoaki Okazaki, Han-Cheol Cho,

Martin Gerner, Illes Solt, Shashank Agarwal, Feifan Liu, Dina Vishnyakova, Patrick Ruch, Martin Romacker, Fabio Rinaldi, Sanmitra Bhattacharya, Padmini Srinivasan, Hongfang Liu, Manabu Torii, Sergio Matos, David Campos, Karin Verspoor, Kevin M. Livingston, and W. John Wilbur. The gene normalization task in biocreative iii. *BMC bioinformatics*, 12(8):S2, 2011.

[23] Yifan Peng, Anthony Rios, Ramakanth Kavuluru, and Zhiyong Lu. Chemical-protein relation extraction with ensembles of svm, cnn, and rnn models. *arXiv preprint arXiv:1802.01255*, 2018.

[24] Martin Pérez-Pérez, Obdulia Rabal, Gael Pérez-Rodríguez, Miguel Vazquez, Florentino Fdez-Riverola, Julen Oyarzabal, Alfonso Valencia, Anália Lourenço, and Martin Krallinger. Evaluation of chemical and gene/protein entity recognition systems at biocreative v. 5: the cemp and gpro patents tracks. 2017.

[25] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[26] Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. Cross-type biomedical named entity recognition with deep multi-task learning. *arXiv preprint arXiv:1801.09851*, 2018.

[27] Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. Gnormplus: an integrative approach for tagging genes, gene families, and protein domains. *BioMed research international*, 2015, 2015.

[28] Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[29] Yaoyun Zhang, Jun Xu, Hui Chen, Jingqi Wang, Yonghui Wu, Manu Prakasam, and Hua Xu. Chemical named entity recognition in patents by domain knowledge and unsupervised feature learning. *Database*, 2016, 4 2016.