Named Entity Recognition and Linking in the Biological Domain

by

Maxwell A. Wenzel

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Bachelor of Science in Computer Science

Department of Computer Science

2020

This thesis entitled:
Named Entity Recognition and Linking in the Biological Domain
written by Maxwell A. Wenzel
has been approved for the Department of Computer Science

_____

Prof. James Martin

_____

Prof. Daniel Larremore

_____

Prof. Chenhao Tan

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Wenzel, Maxwell A. (BS, Computer Science)

Named Entity Recognition and Linking in the Biological Domain

Thesis directed by Prof. James Martin

This paper explores various approaches to building separate named entity recognition and labeling techniques. After experimentation, models that take advantage of tokenizing using RoBERTa were found to significantly outperform conditional random fields models that utilize hand selected feature sets at a multi-class tagging task. A full pipeline performing NER that feeds into NEL powered by RoBERTa models is shown to be highly effective. This shows clear promise in the process of identifying and then Wikifying obscure entities by using a RoBERTa powered NER to NEL pipeline. A pipeline similar to this one could likely be extended to other domains with minimal changes necessary given hand tagged data is provided. All code and data is available at `https://github.com/Max-Wenzel/Bio_NER_NEL`.

Acknowledgements

Thank you to Professor James Martin for acting as my advisor for this thesis. Thank you as well to Daniel Larremore, Marther Palmer, and Chenhao Tan for all volunteering there time to be on my thesis committee during these uncertain times. Thank you to Kate Baldwin, Emma Wenzel, and Aaron Aaeng for proof reading and general assistance. Thank you to Ruth Fernandes for her constant support.

Contents

Chapter

Tables

Table

Chapter 1

Introduction

1.1 Overview

The goal of this project is to create a model that performs Named Entity Recognition as well as Named Entity Linking (which will be from here on referred to as NER and NEL) on texts in a specific domain of environmental research. This will involve the creation of several models each using a different approach. These models can then be evaluated to see which best suits this application.

1.1.1 Purpose

There have been many applications of NER and NEL in the medical domain such as normalizing gene and protein names automatically [4] as well as easing the general analysis of medical texts [1]. However, very little work has been done in creating similar tools for the domains within environmental research. Therefore, I have decided to delve into this as my area of research as there is less work for me to build off of. This allows me to take a fresh approach to comparing and contrasting between a specific and general approach. In this case I will be specifically attempting to link certain entities to a knowledge base where the normalization process will be referred to as linking [7]. This allows for my thesis to be both more challenging and interesting as I am not just reproducing work done before. A tool that could accurately perform NER and NEL on texts in these domains could make an impact in helping facilitate research by removing the need to have more background knowledge or to search for the meaning of a named entity that is unclear to the

reader.

1.2 Related and Previous Work

As mentioned above, there is not much research on NER and NEL specifically in environmental domains. Much of the project will be dedicated to adapting already developed techniques for NER and NEL for these domains.

1.2.1 Named Entity Recognition

This is the process of using a model to automatically find and label named entities found in a text. Named entities can be things such as names, places, times, events, and documents [12]. This is often done by extracting a set of features from each word in a document that is tailored specifically for the domain in which the processing is being done. These features are then used to train a classifier using already labeled texts. These features are not necessarily domain specific, but creating domain specific features can increase the accuracy of recognition at the cost of limiting the model's ability to adapt to another domain. Named entity recognition can be done in two steps where the first is to find any and all named entities in the text. The next pass labels each entity found in the previous pass with its class. A common approach to this problem is to use a bi-directional LSTM [3]. Named entity recognition is very useful as a way of generating further data related to a text to allow for more advanced analysis to be performed on the text.

1.2.2 Named Entity Linking

Named entity linking is the process of taking entities that have been found in a text and linking them to a knowledge base. This can be thought of as NER being the process of finding the named entities and NEL being the process of determining which named entities they are. An important step in this process is to disambiguate the meaning of an entity that could have multiple meanings. A common approach to this problem is that of wikification [17], which is the process of linking named entities to their corresponding Wikipedia page. This can be done by classifying based

on features such as surrounding entities and possible co-references [12], which can help disambiguate the meaning of an entity.

1.2.3 Joint Model of Named Entity Recognition and Named Entity Linking

An important component of both NER and NEL is leveraging the context of a word or phrase in either finding more entities or normalizing them. This includes information such as whether the surrounding words are part of a name entity, what type of named entity they are, as well as which named entities they are. Due to this, research has found that while NER is very helpful in performing NEL, the outcome of NEL can also be helpful in furthering the accuracy of NER [6]. This research has concluded that by not using a joint model for NER and NEL, which can perform both tasks simultaneously, valuable information is being left out that could increase the accuracy of the results. Durrett and Klein found that using a joint model over a sequential independent one led to F-Score increases of up to 1.4 [6].

1.2.4 Feature Extraction

Some approaches such as using conditional random fields or a bi-directional LSTM would require a set of hand extracted and chosen features. These features can be tuned to be specific to the domain and allow for some amount of trial and error as well as informed testing and pruning of the features set. However, another approach worth pursuing is to experiment with language representation models such as BERT [5], which are pre-trained to convert words into vectors and in a way automatically extract context specific features that are ambiguous to domain.

BERT BERT is a group of models that were released by Google as a way of tokenizing words [5]. These models essentially can take in any grouping of words and break it down by using a trained bi-directional model to obtain a list of floating point values for any word within its large vocabulary or break down unknown words into smaller known morphemes. The result is high dimensional vectors that can be easily used for a variety of NLP tasks such as NER. A use

feature of these vectors is that the model was trained with the context of a given word's usage. The result is that words that have similar meanings and uses appear closer to each other in the high dimensional space the vectors exist in. The result is that these features tend to be more descriptive than hand selected ones, at the expense of the models being computationally intense and taking up a significant amount of memory.

RoBERTa This is a variant on BERT as it is a Robustly Optimized BERT pre-training Approach. RoBERTa was released by researchers at Facebook who discovered that the originally released BERT was not trained to its full potential [14]. They retrained the models and experimented with modifying the hyper-parameters and the training tasks in order to better stretch the potential of the models. The result of their efforts is a set of pre-trained models that in many cases exceed the performance that came at the cost of increased training time and size of the models.

1.3 Data

The data being used will be pre-processed academic texts in the environmental domain. The specific domain I plan to have the end model built for is on sea ice; however, this data is not yet available so I will begin the project using biological and earthquake based texts. These annotated data sets have been obtained from the Clear Earth Project which has the goal of advancing NLP in the Environmental and Earth Science domain. This data has already been made available and separated into a development, train, and test set. The annotations of the data are for NER and the training data does not have any sort of linking already incorporated. I will need to further annotate the data to perform NEL. Having the data already well formatted and annotated will greatly speed up my ability to create and test new models on the data. The relatively small size of these data sets will pose challenges in using certain approaches.

1.3.1 Format

The format of these data sets is the CoNLL format which has been labeled using categorized IOB tagging. The CoNLL format is simply splitting the original texts into having each word or punctuation on its own line. Then there is annotation of each of these entries tab separated on the same line. This format is also conducive to adding and storing additional features due to its simplicity. IOB tagging is composed of O tags which indicate a word or character that does not make up a named entity as well as an I and B tag for each category. This allows NER to simply be a classification task with $c * 2 + 1$ classes where c is the number of categories the entities can fit into.

1.4 Approach

The core of the project will be to produce multiple working models that perform NER and NEL on a data set and determine if we can get a significant improvement over another.

1.4.1 Sequential

The first component of the project consists of producing a working model that can perform accurate NER on the texts in the target domain. This first approach is using a bi-directional LSTM and a tailored set of features extracted from the texts. The tested features include part of speech, previous and next word, word stem, and capitalization, among others. Once that has been accomplished I then moved on to producing a similar model that is instead used to perform NEL on the texts. This overall model has a pipeline structure where the output of the recognition model then feeds as the input into the linking model. A significant challenge of this part of the project is that there is no already existing format or annotations to follow for training the NEL model. I need to produce some sort of knowledge base or dictionary to use to normalize the named entities found in the text. This makes verifying the results a greater challenge than with NER; I started this process by generating a proper training set for the linking. For the sake of simplicity and ease

of generating this training set I initially have the goal of performing wikification on the entities and use per-existing libraries that ease this process.

Unified vs Sequential NER For the NER portion of a sequential model, there are further choices to be made on how the classification performed. In much the same way that a full pipeline can be done in a sequential fashion, it is possible to perform the NER in a sequential way. This involves first performing the simple IOB labeling of the data and then, after having this completed, move on to the classes. These classes are assigned to the I and B tags separately from the first step which would reduce the total number of classes being assigned in each step and also reduce the number of entities to be assessed in the second step. It is worth investigating which of these approaches works better for this domain.

1.5 Goals

It is difficult to determine hard set base lines of accuracy for things such as F-Score on our test sets as a "good" score depends on the data set at hand. The end goal is then instead put in terms of producing the two models explained above and have them both be performing the best they can so that I can then determine the better approach and see what sort of increases can be achieved. For the linking I will attempt to properly link named entities that appear to their corresponding Wikipedia articles. The end product would then be both of these models presented in an easy to use and understand fashion and made publicly available alongside an analysis of their performance. As for performance, I would hope to get at least an F1 of .75 with the sequential model and a score higher than that with the joint model although for the reasons stated previously this may change.

1.5.1 On Linking and Moving on to a Joint Model

While there are many resources for implementing a simple NER model, there are far less for NEL. There is also far more work to be done to prepare the data for linking training and possible

wikification. Due to this, I anticipate I may run into some difficulties with this step. While I believe that I should be able to overcome this challenge, in the case that it proves more difficult than expected I may have to cut back on my end goal and reduce it to just a fully implemented sequential model. In the event of these cutbacks, I will greatly increase my expected level of accuracy and will deliver a more advanced and fleshed out sequential model of NER and NEL.

Chapter 2

Named Entity Recognition

2.1 Methods

2.1.1 Using the seqlearn and sklearn-crfsuite libraries

My first attempt at creating a basic name entity recognition model for this data was using a Python package I had used before called seqlearn [9]. While I was able to create a functioning model that was producing results for simple IOB tagging, I ran into several problems that led me to pursue an alternative approach. The package excels at fairly simple sequence labeling tasks and lacks the level of customization afforded by other libraries. This combined with the library using deprecated functions and having less than desirable performance led me to turn elsewhere for a solution.

From this point on I decided to use a conditional random field model for the first attempt at named entity recognition similar to the one created by Lample et al. [13]. For this I chose to use the sci-kit learn and the accompanying sklearn-crfsuite that brings conditional random field based algorithms to the sci-kit learn interface [10].

2.1.2 Data Cleaning

Despite the data having been previously prepared for use, there was some data cleansing that needed to be done. Upon first loading the data, I created a check to make sure all of the tags seen were within the set of tags that were meant to be found in the data. If a label was found to have a

spelling error or missing entirely it was replaced with an O as that would minimize the chance of the label being miss-identified. I also found that a single file was in both the test and train data sets. To resolve this I simply removed the problematic file from the test set as I felt that it was more valuable to have that file in the training data than in the testing data.

As another component of the data cleansing, I compiled the available data in a way that would make it easier to work with. For each of the three directories (train, dev, and test) I performed the aforementioned cleaning on each file in the directory and then concatenated the sentences together by adding a new token `_S_B_` which indicates a sentence break. At this step I also used NLTK's [2] part of speech tagger on each sentence and then stored each words' part of speech as the second column in the data. These sentences with part of speech information provided were then compiled into one master document for its respective source directory. The result was all of the train, development, or testing data being stored in a single file for ease of use.

2.1.3 Conditional Random Field Approach

For this model I used the previously mentioned sklearn-crfsuite which implements a Conditional Random Field approach to classification. This utilizes an undirected graphical model which allows it to learn about the features of the contextual information surrounding each entity to further increase the overall accuracy of the model. For this approach I used limited-memory BFGS with a hand-curated set of features.

Unified vs. Sequential Approach As mentioned previously, I made two different models for named entity recognition. This is implemented by the unified approach being a single CRF model that treats each tag and IOB pair to be a unique class that can be assigned. In the sequential approach there are two separate CRF models that are arranged into a pipeline. The first model is only trained to label each entity with one of the basic IOB labels. The second model takes the output of the first as its input, then the model uses the IOB labels as additional information to help decide which of the sub-categories is appropriate. The idea is that this could possibly lead to

an increase in accuracy as the first step only needs to do the simple labeling and the second step has more information to decide on the sub-category.

Feature Selection As a starting point for the set of features that I would use for this NER model, I consulted the sklearn-crfsuite [10] documentation which included an example starting point for a feature dictionary. The features that I started with and my final set of features can be found in appendix A. As can be seen between the two, the changes that I made to the actual features collected were very minor. The set of features that were provided within the documentation were all general and not biased towards any one set of data. I had originally planned to add in more topic specific features that would be targeted towards helping identify certain tagged entities. For example, I could check if a given word had a common ending found in scientific names "-ia" to possibly increase my accuracy in tagging certain biotic entities. However, in my attempts to use such features, I found that this often had negligible to negative effects on the overall performance of the model. This was compounded by my desire to stick to a less topic specific model so that if it were to be used again in another domain it would not need as many tweaks from the form used for this topic.

The only additional feature that I tested that seemed to cause a significant increase in the performance of the model was a stemmed version of each current, previous, and next word. I did this by using NLTK's implementation of the Porter stemming algorithm. The purpose of this is to remove any information that may not be useful for the evaluation of that word by reducing to its "stem." This involves removing most suffixes that may be indicating time or plurality as these often do not have sway on the tag that a word receives. I also implemented two new optional arguments along with some conditional statements to account for various situations that arise while implementing the sequential approach to the NER step.

Hyperparameter Tuning

The next step in finalizing the conditional random field models is to fine tune the two available hyperparameters for this implementation of limited-memory BFGS, `c1` and `c2`, to maximally increase the performance on the development data. I used sklearn's `RandomizedSearchCV` [19] to test several hundred possible values, plus in this case random combinations of the two values, and choose the ones that performed the best. Doing this process produced a significant increase on the scoring for the development data of around .05 increase in F1 score. I first did this for both the unified and sequential models and found that only the values found when tuning the unified model produced a significant result. Therefore, I decided to use the values found from both steps of the sequential model.

Results After training both the unified and the sequential set ups using the selected features and tuned hyperparameters, I evaluated each of them on three different metrics. All of these are using sklearn's implementation of the F1 score and only differ on which portion of the tag is being assessed. I decided to use F1 score as my metric as I believed that both precision and recall were equally important for this task. It is quite important that the phrases to all be found correctly as well as to not have too many false positives. False positives in the linking step would cause more things to be annotated in a way they shouldn't be, therefore only further complicating the text.

The first F1 score is measuring the accuracy of the model on how it does getting the IOB component of the tag correct. The second is an assessment of how well the model performs on identifying the sub-class of the I and B tags. Finally, there is the overall score which assesses how well the model performs on both of the previous components. It is important to note that as with all of my evaluations of named entity recognition, I am computing the F1 score based on all tags excluding the O tags as they are so frequent that they can skew the score to seem better than it actually is.

Table 2.1: CRF NER — This table shows the F1 Scores on both sets of data. Here Uni refers to the unified model and Seq refers to the sequential pipeline. All values have been rounded to the nearest third decimal.

	Development		**Test**	
Metric	Uni	Seq	Uni	Seq
IOB	0.854	0.845	0.852	0.853
Class	0.806	0.792	0.817	0.821
Overall	0.774	0.755	0.772	0.772

As can be seen above, the differences between the two approaches are negligible. The difference between any pair of the scores between the unified and sequential approach is close to zero with the only exception being the unified approach scoring fairly above the score given by the sequential model.

Verdict After going over the results, I decided to move forward with the unified approach for named entity recognition. The main reason for this is that there is no measurable improvement when using this type of model in a sequential fashion over the unified approach. In addition to the negligible difference in results, the implementation for the sequential approach was more complicated to implement. This then led to an increased number of edge cases where bugs would arise and further complicate getting results. This led me to decide that I should no longer move forward using the sequential model and instead focus only only using the unified model.

2.1.4 RoBERTa Approach

This approach to the named entity recognition step is one that I actually didn't attempt until far into the process. Chronologically I approached this method after implementing a CRF approach to named entity linking as well as a RoBERTa approach to NEL. The results from that inspired me to go back and try this approach as well.

As mentioned in the introduction, I decided to use RoBERTa over normal BERT as RoBERTa showed significant improvements in accuracy [14]. In order to implement this model I decided first to go with Hugging Face's Transformers library [20]. The library, when used along select machine

learning libraries such as PyTorch [18], allows for the use of various pre-trained models such as BERT, RoBERTa, and GPT-2. Through this integration, Hugging Face enables the user to take advantage of these powerful models in tasks such as machine translation, sentence classification, next word prediction, and sequence labeling tasks such as NER. While the Transformers package is expansive and is full of options for the configuration of many types of models, it is more in-depth than is needed for this application. For this reason I decided to turn to the Simple Transformers [11] library.

The Simple Transformers library is built on top of Hugging Face's Transformers model and allows for much faster access to simpler models such as basic named entity recognition and sentence classification. A convenience of the library is that it automatically sets nearly all of the parameters for each model to a convenient default value. It also has features that make it extremely simple to save and then later load a model so that the user does not have to train it each time or go through extensive setup to get this working. As previously stated, one of the models that can be used is an implementation of named entity recognition where the user can use many of the pre-trained models included with Transformers as the tokenizer. I used this model alongside the included pre-trained RoBERTa model as explained previously; then all that was needed was to reformat and label the data.

Data Preparation Unlike the conditional random field model, which uses the CoNLL tab-separated format, the Simple Transformers' NER model takes a different format. Instead it takes a Pandas DataFrame [16] where the first column is a sentence identifier, the second column is the word, and the third column is the tag for that word. This involved using the added sentence breaks to chunk the data into sentences and then assigning each of the sentences an identifier.

Training For the training process I used the default parameters for everything except the saving path and the number of epochs. I found that three epochs was a sufficient number before reaching greatly diminished returns in accuracy. As for the saving path, I modified this for better

organization of directories when running both named entity recognition and linking models so that both can be loaded for further use.

Results As here I am only evaluating a single model, I will not be calculating the scores for the IOB, class, and overall separately. Instead I'm showing the F1 score alongside the precision and recall. For the evaluation I used the built in function that is part of the model class provided for named entity recognition in Simple Transformers.

Table 2.2: RoBERTa NER — This table shows the Precision, Recall, and F1 from the RoBERTa NER model for the development and the test data. All values have been rounded to the nearest third decimal.

Metric	**Development**	**Test**
Precision	0.802	0.821
Recall	0.831	0.848
F1	0.816	0.834

2.2 Final Results

From looking at the results from each approach, it seems clear that using a RoBERTa powered model is significantly superior to the conventional conditional random fields approach. Comparing the overall F1 scores result in a 0.043 improvement for the development data and a 0.62 improvement on the test data set. These are fairly large improvements and bring the overall score over my F1 goal score of 0.800. Another advantage of the RoBERTa approach is the ability to save the model easily and then load it for additional testing. With the CRF model, I was required to retrain the data each time I wished to perform additional testing. Yet another advantage of not using the conditional random fields approach is that by using RoBERTa there are no hand selected features and therefore this model can easily be extended to any domain simply by changing the labels and data inputted.

Chapter 3

Named Entity Linking

3.1 Data Labeling

The primary part of the process of named entity linking is identifying which entities should be linked. There will be some entities that would not provide any extra information if they were to be linked. The first step in determining this was to examine each class that is labeled by NER and see if it is worth linking. The next step then is to determine the subset of each of those classes of interest that the reader would gain information from by linking to a database such as Wikipedia.

The purpose of the linking is to find obscure or likely unknown terms and then link those to a place where the reader can obtain more information about the subject. This means that for each class of interest this model needs to identify the portion of those labels that are for an entity that is most likely not known by an average person reading the given text.

3.1.1 Labels

This section gives a brief overview of each label and whether it may be of interest. I obtained this information by examining a random distribution of the contents of each class.

Biotic Entity This is any given entity ranging from specific species such as "chrysothamnus viscidiflorus" to general biotic concepts such as "body". A good number of these would likely have their own Wikipedia entry. Due to this, I believe that entities of this type should be considered of

interest as there is a potential for many of these to be obscure.

Eventuality These are outcomes and effects that are the result of some phenomenon such as states like "decomposition" or actions that are caused by a phenomenon such as "escapes." A majority of these are fairly common words with only a few more obscure outliers. This could possibly be of interest and may be worth considering for future work.

Aggregate Biotic Abiotic Entity These are regions where both biotic and abiotic entities exist such as "mountainous areas" and "mangrove forests." These seem to be made up almost entirely of combinations of common adjectives and verbs. Due to the meaning of nearly all of these labels likely being apparent to most readers, I do not consider this label to be of interest.

Value This is as simple as it sounds – a value that is describing some sort of measurement such as "90.5" or "tens of thousands." This tag is not of interest due to all of the entries of this relatively small set being well known to most people. There is also no informative way to link these values that would provide additional information the entity itself doesn't already give.

Quality These are terms that are used to describe either the biotic, abiotic, or aggregate entities. Examples include "erratic" and "native." While most of these descriptors are fairly general, there are a few that are domain specific. Therefore this class could possibly be of interest.

Time

These describe specific times, durations, intervals, and time relevant events such as "growing season," "for some time," and "1904." Due to the consistent normality of these entities, they are not of interest in regards to linking.

Unit This goes along with value. Where value is the value of a measurement, unit is the unit of a measurement such as "centimeters" or "%." This class is also not of interest for the same

reasons that value is not.

Abiotic Entity These are all of the significant entities that are abiotic such as "copper," "cracks," and "sand." Some of the names present in this category are more obscure or include scientific names that may be not known by the average reader which makes these labels of interest.

Location These are also quite simple in that they are just locations such as "North America" and "central Utah." Due to the tendency of these entities to be on the more general side and often using common locations, I do not think that there would be much value in linking some of these entities.

3.1.2 Labeling Process

A major constraint on producing a named entity linking model with this data is that it has only been annotated and tagged for the purposes of named entity recognition. This means for each class I wish to perform named entity linking on I need to hand label myself. For this reason I chose to only perform named entity linking on a single class that appeared to be the most promising for this application. In this case this class was the Biotic Entity label as it contained many instances that have potential for linking.

The largest offender in this category is by and large the scientific species names for various plants and animals. Another category that should be identified by this classifier is biotic entities that are not scientific names but are still rather uncommon and unlikely to be known by the average person such as "sea squirts." Lastly are relatively uncommon biotic terms such as "pedicel," which is a stem that connects to a single flower. I have found the most effective way to link these is through simple wikification as many of these have their own Wikipedia article. This will require going through and giving the terms their own tags to indicate they need to be linked. In order to make labeling as painless as possible, I will be appending `-L` to all `B-Biotic_Entity` tags that I believe should be linked. This simple labeling schema allows me to search through all of the biotic

entity beginning tags and append this to any of them I believe need to be linked.

In order to make sure this was promising, I went through some of the entities I wished to link and ran them through the Wikipedia library [8] for python that gives access to the Wikipedia API. One difficulty I have found is scientific names where the first part is abbreviated are hard for Wikipedia to disambiguate e.g. E. Coli should direct Escherichia Coli, but Wikipedia does not always make this correct jump. For this I will have to assume that the given text has followed the convention of mentioning the full version previously and therefore the information will be provided for the reader at that place.

Other than the previously mentioned issue, I went through and appended the tag to all of the entities I deemed appropriate. This required me to have a set of consistent mental rules in making the call if an entity was sufficiently obscure that it should be linked. I followed a general rule of tagging any entity that I did not know the definition of and for those I knew vaguely or knew to not be common knowledge. Luckily for me, I am not particularly well versed in biological terms or biology in general which allowed me to have a close to unbiased view on many of these entities. All in all, it took me roughly ten hours of hand tagging to fully tag the biotic entities in the three sets of data.

3.2 Methods

It is worth mentioning that at this point after tagging the data, I had to go back and modify some of my previous work in order to maintain functionality. This simply consisted of modifying the functions for loading data for the purpose of named entity recognition so that they would slice the `-L` off any biotic entities where it is present.

3.2.1 Conditional Random Fields Approach

The first way I wanted to address the issue of this additional labeling step was to simply treat this biotic entity with the appended tag as another class. In other words I just added `B-Biotic_Entity-L` to my list of possible tags for my conditional random field named entity

recognition model. My reasoning behind this is that this problem could potentially be treated as a simple extension of the NER step. It is also worth checking to see if it was an effective strategy as it would make the entire process of named entity linking much easier. This was a very simple adaptation as I simply disabled the filter for removing that appended tag in the loading data step and added that new longer tag to my data.

After doing this minor modification, I retrained the model on this set of data and performed the same evaluation process as was done for the NER step. After doing this I did notice a very slight decrease of around 0.012 for the overall F1 score of the model on both sets of data. One of the parameters of the scoring function included with sklearn-crfsuite is the labels that are to have an F1 score computed. By passing a single tag to the evaluate function the score is essentially reporting how well the model was at finding and labeling that specific tag. Upon doing so I was able to get an overall score of 0.714 and 0.669. These are both significantly below the overall scores with this tag added as well as the overall scores from the previous NER CRF model. This shows that this approach is not particularly good at identifying the linked entities.

Limited Tag Approach Something I was interested in testing was if there was a way to increase the scoring on the named entity linking tags by removing some of the tags that I knew would not be used for NEL. To try this I filtered out the tags for Time, Location, Value, and Unit when I loaded in the data and then repeated the training and evaluation process. After doing this the results showed a slight increase in the overall score, but it also showed a decrease in the score for the `B-Biotic_Entity-L` tag in the development data and a slight increase in the score for the test data. This approach as a whole seems to increase the accuracy very slightly by giving the model less options to choose from; however, I think doing this removes some contextual information that increases the score in certain cases. The result overall is that this seems to not be a good modification as it produces negligible change overall in the scoring in the overall and specific NEL

sense.

Results After seeing the results of treating the significant biotic entities as a separate class, it seems that this approach is not quite sufficient for the task at hand. This approach falsely treats the biotic entities of interest and those that are not as being two distinct classes. This does not properly account for the fact that the -L tagged entities are still just as much biotic entities as the non -L tagged ones. In other words, by changing some of the biotic entity beginning tags into another tag entirely only hampers the named entity recognition step's ability to perform what it was meant to do, the named entity recognition.

3.2.2 RoBERTa Approach

After experimenting with a unified approach to linking, I decided that this model would benefit most from a sequential set up. It seems best in this approach to treat the identifying labels that should be linked as a separate process. While consulting the various model types offered by Simple Transformers I found sentence classification seemed to have promise for this application.

Phrase Classification While this model is often used for sentence classification, a much more apt name for this approach would be binary phrase classification. The problem can be addressed in this light as in the data each tag that starts with `B-Biotic_Entity` is either ended with an `-L` or it isn't; therefore, it can be said that any biotic entity that has the appended tag should be classified as a 1. If it is not to be linked then it should be classified as a 0.

The easiest way to do this would be to take sentence as a time and treat them as a 1 if they contain a biotic entity that should be linked and a 0 if they don't. This of course though is not a valid or very good approach to this problem. It is not unreasonable to imagine a situation where there are multiple biotic entities and some of them should be linked and some shouldn't. The improved approach that I decided to go with is having each biotic entity found in the data have its own phrase. This can be done by having all parts of a biotic entity become the core of

the phrase. This core would be made up of the beginning/B tag of type biotic entity and then every following `I-Biotic_Entity` tag until any other tag is found. The next step is to include the preceding and following words around the entity to be classified. This component is important as it provides some context to the entity that could potentially be useful in classifying the entity.

It was then necessary to create another helper function that could be fed the normally formatted data and extract these phrases. This function also returned the phrase with either a 1 or a 0 depending on whether the core of the function originally was annotated with a `-L`. Another part of this function was to return a span made up of two integers that represented the position of the words that began and ended the phrase in the entire set of data. The point of these spans was to make calculating a score for this step easier. After that, all that was left was to use Simple Transformer's `ClassificationModel` with the same parameters used in the NER model.

Results Using the built-in evaluation function of the classification model, I was able to obtain F1 scores for the development and test data when using the perfect phrases, meaning this evaluation is using all of the biotic entities as they are labeled in the data rather than found by an NER model.

Table 3.1: RoBERTa NEL — This table shows F1 score from the RoBERTa NEL model for the development and the test data. The Diff value is the improvement over the score achieved by the CRF approach to NEL on just the beginning biotic entity tags. All values have been rounded to the nearest third decimal.

Metric	**Development**	**Test**
F1	0.846	0.850
Diff	0.132	0.181

Due to the named entity recognition model being multi-class and not binary, the difference row should not be seen as an exact direct comparison as it should naturally be expected that the binary model will have a higher score. Despite that, this score is very impressive and well exceeds the performance I was looking for.

3.3 Final Results

As can be see from the RoBERTa model's results, they far exceed the performance of the conditional random field approach. Additionally, there are the previously mentioned issues that are added by including the named entity linking tag in with the other tags used for the NER step. For these reasons moving forward to the end product I will be using the RoBERTa NER and phrase classification models. While these results are good, they do not quite represent how well this model will perform once all of the pieces have been put together. These results are the product of the NEL model being fed perfectly spanned phrases that are exactly where the biotic entities should be. In the final product there will be no guarantee that these phrases will be the same as those built from the correct tags.

Chapter 4

Full Pipeline

The full pipeline setup is the final product of my thesis. The purpose of the final pipeline is to demonstrate the full functionality of a model that can use two pre-trained models to perform named entity recognition and then named entity linking in full. In this case the direct predictions taken from the NER model are being fed into the NEL model, which then classifies the biotic entities which will then be appropriately Wikified.

4.1 Methods

Due to the way I constructed the infrastructure around both of the RoBERTa based models, it was fairly simple to combine them together to form a singular pipeline. The first step of the pipeline attempts to load the named entity recognition model, or if it hasn't been trained yet, to train it. The evaluate function that I created is then called as it stores a list of predicted tags for both of the data sets. The next step creates a copy of the original data from the development and test sets.

The next steps are several for loops that loop over the original data and replace the actual tags with the predicted tags. Although it seems strange to create this data by converting back to the original format, this makes feeding into the named entity linking model trivial as I already had created a function to convert the original format into the phrases and spans format. The pipeline then checks for a pre-trained NEL model or trains it if it has not been trained previously. Finally the pipeline uses the model to predict the classification of all the phrases that were found by the

NER model.

4.1.1 Wikification

In order to perform the Wikification I used the previously mentioned Wikipedia [8] Python library that allows me to query Wikipedia. I did this by creating a list that has an entry for each found phrase. Next I used the search function of the library on the core entity portion of the phrase and passed in an argument to return only the top result which returns the name of the page most related to the search term. Then this top result is passed into the page function to get the url of the corresponding Wikipedia page. The reasoning behind this is that not every entity to be linked will have a Wikipedia entry of its own; this way it gets the most relevant page in the event an exact matching page is not found. This is still useful as the reader could use this related page to find more information about the entity or gain more context for the entity.

4.1.2 Evaluation

Something that must be considered in evaluating the full pipeline is how errors in the NER step can cascade and cause complications in the NEL step. These errors can manifest in a few different ways. One of these is that the NER model can falsely identify a biotic entity as being larger than it is; this will cause the resulting phrase to contain extra context than necessary. This can be grouped into a larger category of the spans of biotic entities being slightly off.

There is also the potential for an error in the number of entities found. This at its most basic is just a biotic entity being found in a group of words where it shouldn't be. For example, an abiotic entity falsely being thought to be a biotic one. There also could be an instance where a single biotic entity is instead identified to be a grouping of biotic entities resulting in extra biotic entities that didn't exist in the actual tags. Lastly, the inverse could happen where multiple biotic entities could be falsely grouped into a single biotic entity and thereby decrease the number of entities found relative to the actual amount.

The problem here is that this makes it impossible to accurately score the performance in the

same way we scored the performance of the NEL model when using perfect spans. The only way to safely score any of the imperfect spans would be if they happened to have the exact same contents as an actual span. However, due to the errors listed above, it would be very likely that many spans would not properly overlap with the correct spans.

Solution The solution I found to this problem is to essentially undo the step of turning the data into phrases. To do this the spans of the phrases that were found to create an output that is more similar the output of the NER mode can be used. The first step to this solution is to create a list of zeros that has a length slightly larger than the last span. Second loop through all of the spans and their corresponding predictions from the NEL model. For each span, if the prediction was that the phrase should be linked, then change all of the entries that have an index within that span of values to a 1. Once this has been done to every span then return the list.

After the function that does this is run on both the actual data and the predictions, then append the appropriate number of zeros to one of the lists if one is shorter than the other. Then it is possible to calculate the F1 score of the results as usual on these new extended sequences. The difference here being that when there is partial overlap in two spans that should be linked then the amount of score rewarded is proportional to the amount of overlap; this reflects how being a word or two off could still lead to an informative linking.

4.2 Results

Using sklearn's F1 score function on the transformed outputs I was able to receive scores that more accurately reflected the performance of the two steps in tandem.

Table 4.1: Full Pipeline NEL — This table shows F1 score from the RoBERTa NEL model using the RoBERTa NER model output as input for the development and the test data. The Diff value is the improvement over the score achieved by the CRF approach to NEL on just the beginning biotic entity tags. All values have been rounded to the nearest third decimal

Metric	**Development**	**Test**
F1	0.797	0.808
Diff	0.083	0.139

These results are indicative of how the RoBERTa NEL model is performing in the full pipeline setting. Despite the imperfections in the name entity recognition step, the decreases in accuracy are not particularly large. The performance is very good and meets my best expectations goal of approximately an F1 score of 0.8.

Chapter 5

Conclusion

I have found through my experimentation that the best solution for implementing both named entity recognition and linking on this set of biological text data is to use a state of the art pre-trained model such as RoBERTa. The use of this model demonstrates how the abstracted features gained through tokenization are easily able to outperform hand selected and even domain specific features. The models that I have utilized are highly capable of finding the relevant entities within the text and then, given properly annotated data, classifying phrases as either containing an obscure entity or not with relatively high levels of accuracy.

It would likely be worth generating labeled data for some of the other classes that are tagged in the named entity recognition step. This research shows the viability of extending this work into the other domains of data available as these models do not require any sort of domain specific tuning other than that done at the training step.

Bibliography

[1] Asma Ben Abacha and Pierre Zweigenbaum. Medical entity recognition: A comparison of semantic and statistical methods. In Proceedings of BioNLP 2011 Workshop, BioNLP '11, pages 56–64, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[2] Steven Bird, Ewan Klein, and Edward Loper. Natural Language Processing with Python. O'Reilly Media, Inc., 1st edition, 2009.

[3] Jason P.C. Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. Transactions of the Association for Computational Linguistics, 4:357–370, 2016.

[4] Aaron M. Cohen. Unsupervised gene/protein named entity normalization using automatically extracted dictionaries. In Proceedings of the ACL-ISMB Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics, ISMB '05, pages 17–24, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

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

[6] Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. Transactions of the Association for Computational Linguistics, 2:477–490, 2014.

[7] Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. Evaluating entity linking with wikipedia. Artificial Intelligence, 194:130 – 150, 2013. Artificial Intelligence, Wikipedia and Semi-Structured Resources.

[8] https://github.com/goldsmith. Wikipedia. https://github.com/goldsmith/Wikipedia, 2016.

[9] https://github.com/larsmans. seqlearn. https://github.com/larsmans/seqlearn, 2016.

[10] https://github.com/TeamHG Memex. sklearn-crfsuite. https://github.com/TeamHG-Memex/sklearn-crfsuite, 2019.

[11] https://github.com/ThilinaRajapakse. Simple transformers. https://github.com/TeamHG-Memex/sklearn-crfsuite, 2020.

[12] Dan Jurafsky and James H. Martin. Speech and Language Processing. 3rd edition, Draft, 2019.

[13] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. CoRR, abs/1603.01360, 2016.

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[15] Yinxia Lou, Yue Zhang, Tao Qian, Fei Li, Shufeng Xiong, and Donghong Ji. A transition-based joint model for disease named entity recognition and normalization. Bioinformatics, 33(15):2363–2371, 03 2017.

[16] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, Proceedings of the 9th Python in Science Conference, pages 51 – 56, 2010.

[17] Rada Mihalcea and Andras Csomai. Wikify!: Linking documents to encyclopedic knowledge. In Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM '07, pages 233–242, New York, NY, USA, 2007. ACM.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019.

[19] 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, 12:2825–2830, 2011.

[20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. ArXiv, abs/1910.03771, 2019.

Appendix A

CRF Features

A.1 Beginning Features

```
def feature_extractor(sentence, i):
    word = sentence[i][0]
    pos = sentence[i][1]
    features = {
        'bias': 1.0,
        'word.lower()': word.lower(),
        'word[-3:]': word[-3:],
        'word[-2:]': word[-2:],
        'word.isupper()': word.isupper(),
        'word.istitle()': word.istitle(),
        'word.isdigit()': word.isdigit(),
        'pos': pos,
        'pos[:2]': pos[:2],
    }
    if i > 0:
        word1 = sentence[i - 1][0]
        pos1 = sentence[i - 1][1]
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:word.istitle()': word1.istitle(),
            '-1:word.isupper()': word1.isupper(),
            '-1:pos': pos1,
            '-1:pos[:2]': pos1[:2],
        })
    else:
        features['BOS'] = True

    if i < len(sentence) - 1:
        word1 = sentence[i + 1][0]
        pos1 = sentence[i + 1][1]
        features.update({
            '+1:word.lower()': word1.lower(),
            '+1:word.istitle()': word1.istitle(),
            '+1:word.isupper()': word1.isupper(),
            '+1:pos': pos1,
            '+1:pos[:2]': pos1[:2],
        })
    else:
        features['EOS'] = True

    return features
```

A.2 Final Features

```
def feature_extractor(sentence, i, step_two=False, predictions=False):
    word = sentence[i][0]
    pos = sentence[i][1]
    features = {
        'bias': 1.0,
        'word.lower()': word.lower(),
        'ps.stem(word)': ps.stem(word),
        'word[-3:]': word[-3:],
        'word[-2:]': word[-2:],
        'word.isupper()': word.isupper(),
        'word.istitle()': word.istitle(),
        'word.isdigit()': word.isdigit(),
        'pos': pos,
        'pos[:2]': pos[:2],
    }
    if step_two:
        if predictions:
            features.update({
            "IOB": predictions[i]
            })
        else:
            features.update({
                "IOB": sentence[i][2][0]
            })
    if i > 0:
        word1 = sentence[i - 1][0]
        pos1 = sentence[i - 1][1]
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:ps.stem(word)': ps.stem(word1),
            '-1:word.istitle()': word1.istitle(),
            '-1:word.isupper()': word1.isupper(),
            '-1:word.isdigit()': word1.isdigit(),
            '-1:pos': pos1,
            '-1:pos[:2]': pos1[:2],
        })
        if step_two:
            if predictions:
                features.update({
                "-1:IOB": predictions[i - 1]
                })
            else:
                features.update({
                    "-1:IOB": sentence[i - 1][2][0]
                })
    else:
        features['BOS'] = True

    if i < len(sentence) - 1:
        word1 = sentence[i + 1][0]
        pos1 = sentence[i + 1][1]
        features.update({
            '+1:word.lower()': word1.lower(),
            '+1:ps.stem(word)': ps.stem(word1),
            '+1:word.istitle()': word1.istitle(),
            '+1:word.isupper()': word1.isupper(),
```

```
            '+1:word.isdigit()': word1.isdigit(),
            '+1:pos': pos1,
            '+1:pos[:2]': pos1[:2],
        })
        if step_two:
            if predictions:
                features.update({
                    "+1:IOB": predictions[i + 1]
                })
            else:
                features.update({
                    "+1:IOB": sentence[i + 1][2][0]
                })
    else:
        features['EOS'] = True
    return features
```