Initial Project Report

Bill Xia
Tufts University
wxia01@tufts.edu

1. Introduction

Our goal is to develop a prediction model that is able to tag words in biomedical paper abstracts as fragments of non-consumer biomedical terms (which we'll call expert terms going forward) or non-expert terms. Essentially, the model will take a sentence from a biomedical paper abstract as its input and output a list of labels identifying each word in the sentence as either part of an expert term or not part of an expert term.

We believe that probabilistic modeling will be useful for this task because such an approach leverages prior word and label data to predict future labels. This feature of probabilistic models allows us to make use of the context that terms appear in to determine whether they're expert terms or not. Additionally, a probabilistic model offers a less computationally intensive way of leveraging context than more advanced models such as transformers.

2. Data & Analysis Plan

Our data is derived from the PLABA dataset[1], a collection of biomedical paper abstracts. Last summer, I annotated the data by constructing Beginning-Inside-Outside label sequences for each sentence. That is, each word in each sentence was labeled either B (beginning of an expert term), I (inside an expert term), or O (outside an expert term). The following is an example observation-label pair:

```
Quinine sulfate is an antimalarial drug.
B       I       O  O  O            O
```

Inputs to the model take the form of individual sentences (strings) while outputs take the form of label lists, where each label corresponds to a single word in the sentence.

In total, there are 7606 sentence-label-list pairs. There are 3201 sentences (42.09% of the total data) that contain expert terms (B in the label list). Of those sentences, there are 1356 (42.36% of the sentences with expert terms, 17.83% of the total data) that contain expert terms consisting of more than 1 word (I in the label list).

[1] Attal, K., Ondov, B. & Demner-Fushman, D. A dataset for plain language adaptation of biomedical abstracts. Sci Data 10, 8 (2023). https://doi.org/10.1038/s41597-022-01920-3

2.1 Necessary Pre-Processing

The data requires very little pre-processing. Input sentences must be stripped of all punctuation and have all their characters set to lowercase. Additionally, each sentence must be split into lists of words for the model to predict labels for each word one at a time.

2.2 Visualizing the Data

Because our model will predict word labels according to prior words and labels in a given sentence, it is important to understand how much context (that is, how many words) we are shown before expert terms occur in each sentence. The number of words that occur before every expert term will tell us how much information our model can use to make each prediction.

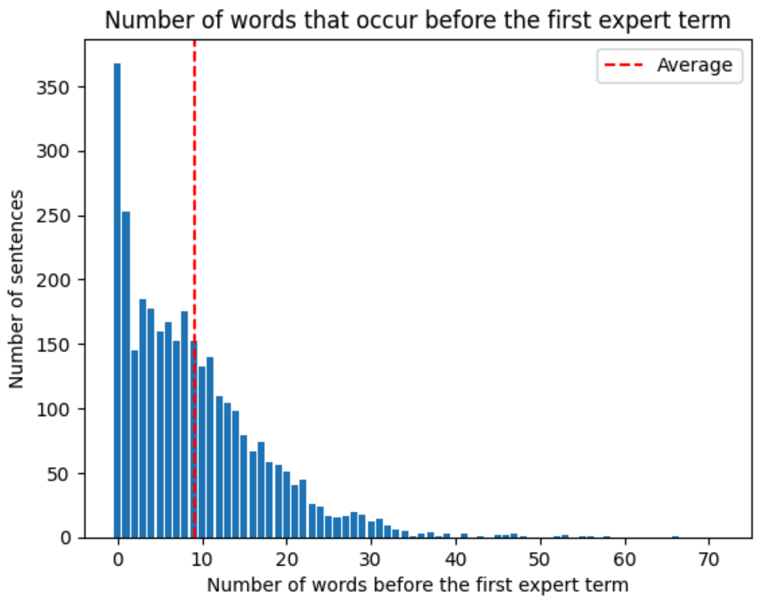

Figure 1. Average number of words that occur before the first expert term in sentences with expert terms.

On average, 9.114 non-expert terms appear before the first expert term, meaning that context can often be used to predict expert terms. However, 368 sentences (11.5%) begin with an expert term, meaning that for a non-significant number of sentences, we won't have any context to predict the first expert term. As such, we must find a way of forming accurate predictions for the labels of words that occur at the very beginning of sentences.

2.3 Experimental Protocol

Our data will be randomly partitioned into training, validation, and testing sets according to an 85 : 5 : 10 ratio. They will have 6465, 380, and 761 sentence-label pairs respectively. We believe that such a split is reasonable because the model will require as much training data as possible to build a substantive vocabulary.

We plan on evaluating the performance of our models based on how often they can correctly label individual words in sentences in our test set. Specifically, we plan on using precision, recall, and F1 score to measure model performance. Precision will measure what percentage of the expert terms labeled by our models are actually expert terms. Recall will measure what percentage of actual expert terms our models are able to catch. Together, they form the F1 score, which measures an average between precision and recall to give a more general metric of our models' performances.

3. Baseline Method

For our baseline model, we will make a Markov assumption and say that each label's value is dependent on the previous label's value. We'll also make use of the current word (the one associated with the predicted label) to make our predictions.

Our model takes the form of a Categorical PMF that predicts random variable l_t given either parameter η or parameter ζ.

- l_t is a discrete integer representing the t-th label in the sentence. It has a sample space of {0, 1, 2}, which has size L=3.
- Both η_{mv} and ζ_v are probability vectors of length L=3. They represent the likelihoods of each possible BIO value that l_t could have. Both are bound by the following constraint:
    - All elements in both vectors must be greater than 0.
    - The elements of the vectors must both sum to 1.
- We use parameter ζ_v when predicting the first label in a given sentence. We choose ζ_v by indexing into ζ (a V * L array) using w_t, the integer representation of the first word in the sentence. ζ maps integer representations of words to probability vectors for the possible labels associated with those words.
- We use parameter η_{mv} when predicting all labels beyond the first in a given sentence. We choose η_{mv} by indexing into η (an L * V * L array) using l_{t-1} and w_t, the integer representations of the previous label and the current word respectively. η_{mv} maps integer representations of prev-label, curr-word combinations to probability vectors for the possible labels associated with those combinations.

So, the likelihood function used to predict the label of the first word in a sentence looks like this:

$$\mathrm{CatPMF}(l_t \mid \zeta_v) = \prod_{l=1}^{L} \zeta_{vl}^{[l_t = l]}$$

The likelihood function used to predict the labels of all other words in a sentence looks like this:

$$\mathrm{CatPMF}(l_t \mid \eta_{mv}) = \prod_{l=1}^{L} \eta_{mvl}^{[l_t = l]}$$

3.1 Estimation

We will perform label predictions using a maximum likelihood method. For each word, we will index into η using l_{t-1} and w_t to get η_{mv}. We'll then use η_{mv} as the parameter for our Categorical likelihood function. Maximizing the likelihood function over all values in the label set will yield our prediction for l_t.

$$l_t = \text{argmax}_{l \in L} \text{CatPMF}(l \mid \eta[l_{t-1}, w_t])$$

3.2 Implementation

Our baseline model is simple enough for us to implement entirely from scratch, using the numpy[2] library's argmax function to perform maximization.

4. Hypothesis-Driven Upgrade Method

One weakness in our baseline is that it does not have a good way of predicting labels for label-word combinations that don't occur in the training data. We can solve this problem in the upgrade by introducing a prior distribution.

Our prior distribution will take the form of a Dirichlet PDF with random variable ζ_v and parameter β for the first prediction in a sentence and random variable η_{mv} and parameter α for all subsequent predictions.

- ζ_v and η_{mv} are the same as described in the baseline section.
- β and α are both vectors of size L with elements equal to 1.005. They essentially represent pseudocounts for how many times each label occurs for a given prev-label, curr-word context window. These pseudocount vectors will help us make more reasonable predictions for context windows not seen in the training data.

With our Categorical likelihood and Dirichlet prior, we can perform predictions using a MAP estimation. We hypothesize that compared to our baseline, the upgraded model should improve prediction accuracy on our dataset, especially when predicting labels given prev-label, curr-word combinations not seen in the training data, because of the upgrade's use of pseudocounts in the Dirichlet prior distribution.

4.1 Implementation

As with the baseline model, our upgrade is simple enough for us to implement it entirely from scratch. As before, the numpy library will supply us with the argmax function, which we'll use for optimizing the objective function.

[2] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). https://doi.org/10.1038/s41586-020-2649-2

5. Timeline

Milestone	Due Date
Pre-process and split data into training, validation, and test sets.	Saturday, 4/20/24
Finish implementing and evaluating the baseline model.	Monday, 4/22/24
Begin implementing the upgraded model; find gaps in my understanding.	Thursday, 4/25/24
Visit OH to fill gaps in my understanding of the upgraded model.	Thursday, 4/25/24
Finish implementing and evaluating the upgraded model.	Thursday, 5/2/24
Submit final report	Thursday, 5/9/24

6. Questions

Question 1

My first question involves how I construct probability vector tables η and ζ. Currently, my plan is to create a table of counts, where every time I encounter a label given a certain context, I increment that label's count at that context's index in the appropriate table. For a simple example, say we're making the count table to build η. In the training data, we observe some state t, where $l_{t-1} = 0$ and $l_t = 1$. If we ignore w_t for the sake of simpler visualization, here is how we would increment the count table:

	$l_t = 0$	$l_t = 1$	$l_t = 2$
$l_{t-1} = 0$	0	+1	0
$l_{t-1} = 1$	0	0	0
$l_{t-1} = 2$	0	0	0

Note that we only increment the cell that corresponds directly to the current context. Here is an increment system that I am considering implementing. Given the same state t as above, here is how the new system would increment the table:

	$l_t = 0$	$l_t = 1$	$l_t = 2$
$l_{t-1} = 0$	0	+2	0
$l_{t-1} = 1$	0	+1	0
$l_{t-1} = 2$	0	+1	0

The main difference is that we now increment all cells that link the current label to the current word, but we increment the cell that links the current label to the previous label more. I believe that the words themselves are a greater indicator of a word’s “expertness” than previous labels. This system would capture this hypothesis by creating stronger associations between words and labels than labels and labels.

My question, then, is whether this new system is worth implementing. It is more complicated than my current system, but I believe it will better capture the relationships between labels and the contexts that they appear in.

Question 2

Another question I had to do with how I’m preprocessing my data. Currently, I’m splitting my sentences into words, which is how they’re labeled. I’m curious to know if splitting them into tokens rather than whole words will improve my model’s ability to perform predictions.

Using tokens instead of words would expand my program’s vocabulary and improve its ability to predict labels for novel contexts, as new words may share fragments with words our program has seen before.

Splitting our data into tokens will come with several drawbacks. First, we’ll have to pre-process the label lists to label tokens instead of words. Secondly, it may become necessary to expand our context window to make accurate predictions, as word fragments don’t carry as much meaning as whole words. Making this adjustment will increase the computational costs of our model.

Is this change worth implementing? I’m unsure about how the benefits weigh against the potential downsides of this change.