CS505 NLP: Milestone 2 - Data and Baseline

Chris Krenz
Boston University
ckrenz@bu.edu

Abstract

In biological research and manufacturing, biologists often submit orders to wet labs for various functions, such as DNA synthesis or sequencing. This interface between the client and service provider can be fraught with various inefficiencies and barriers to entry. To simplify this process and facilitate the client's ability to submit complex biodesign and biomanufacturing orders, I propose a machine learning model to take in natural language prompts to predict the user's desired workflow (i.e. sequence of lab services) and (potentially) the associated parameters. The DAMP Lab at Boston University offers a manageable number of services (and service bundles) from which the model could choose, and the training corpus could be acquired through targeted internet searches of these keywords to find the terminology and phrases that tend to accompany them (for example, references to 'gene editing' or 'modifying sequences' may tend to accompany gene editing services, such as Gibson Assembly and Modular Cloning).

1 Data

The labels for this project were gathered from the DAMP Lab website's listing of available services.[1] While biology labs more broadly can perform a wider range of services, this common set of services will constitute a more manageable number. From this source, I (manually) extracted 38 labels, such as "gibson assembly" and "modular cloning" (though I may expand this list to include variations on the different services). The model will attempt to construct a sequence of these services (i.e. a workflow) based on a natural language description of the desired work order from the user/customer (roughly 1-3 sentences long).

For the time-being I have chosen to assemble the training corpus (the input sentences that relate to each of these services) from PubMed.[2] Articles on PubMed are numerous and relatively easy to access via Pythons' BioPython library, specifically, the NCBI Entrez module, which is an API that allows you to pull article metadata–including abstracts–from PubMed.[3] While I considered trying to pull data from article contents or other sources, the abstracts are far more accessible and easy to manage. From this source, I fetched 9,999 articles (that contain one or more of the keywords/labels identified above), of which, 9,942 had available abstracts. I further decomposed these abstracts into sentences that contain at least one of the keywords/labels, resulting in 2,687 sentences in the training corpus. As a final pre-processing step, I removed extraneous characters and converted all words to lowercase to simplify training using the nltk package.[4]

As one example of a resulting sentence in the training corpus, we have the following:

> *"plasma sample collected normal healthy individual lung cancer patient **protein purification** analysis conducted using lcmsms"*

I have bolded the keyword on which the sentence was fetched and extracted. Of note in this example might be the phrase "plasma sample," which is relatively likely to occur in close proximity to discussions of "protein purification".

Note, as I progress further in the project, I may very well expand the size of this corpus. With the scripts I have setup, this should be relatively easy to accomplish by either 1) expanding the number of keywords (such as by including variants of the existing keywords or expanding to include services offered by other laboratories, or 2) increasing the number of articles I fetch from PubMed. However, at this stage, 2,687 sentences is a manageable number.

A critical remaining step is how to generate labels for the training corpus. I am still evaluating different approaches here, including unsupervised learning, however, at this point in time, I am implementing the simple approach of automatically applying labels to each sentence based on the inclusion of one or more keywords in the sentence. While this can be done simply, it does run the risk of over-fitting on keywords, rather than finding contextual patterns. Still, as a first pass, this approach has proven useful. And I may be able to mitigate the over-fitting problem by including a larger n-gram model to include more context, implementing L2 regularization, or TF-IDF weighting,[5] among other options. I may also need to add to the corpus by including sentences that flank the target sentence but do not include the keywords (or even simply remove the keywords from the sentences). Despite such concerns, this approach to labeling avoids the need for laborious (and infeasible) manual labeling of data.

Another factor to consider is the fact that 'workflows' are technically *sequences* of services, wherein order matters. However, tracking label order would be both more complex and likely unnecessary. For the kinds of short workflows I will be considering (based on 1-3 sentences, with generally only one label generated from each), services tend to occur in very predictable orders. Currently, I plan to apply a simple algorithm at the last step that will put the services in a predetermined order. As such, the model will really be predicting a bag of services, rather than a specific sequence. This is simpler, yet still sufficient for my purposes.

2 Baseline

At the moment I am planning on relying primarily on the f1 score metric, which is a reflection of both precision and recall. I may consider changing or adding metrics as I progress, but at this stage, the goal is a simple metric that will allow me to establish a reasonable benchmark.

The benchmarking approach I decided upon is to use OpenAI's GPT-4 API: providing it the corpus/services and asking it to assign a service to each sentence. This was a simple and accessible implementation that resulted in an overall accuracy of **68%**.

This benchmark is, quite frankly, ambitious. Given the comparative sophistication of GPT-4, I do not anticipate surpassing this target, but it still serves as a useful stretch goal. Even if I fall short, hopefully I can at least do better than chance!

3 Conclusion

I have now established my source of data, fetched some data, cleaned it, and benchmarked it. I am now continuing to experiment with the approach and model I want to develop. For my implementation, I am exploring scikit-learn's TF-IDF Vectorizer and Logistic Regression modules as a useful frame of reference.[6] I plan to split the corpus into two sets: 80% for training and 20% for testing. While a lot of work still remains, I've made fruitful progress and am (hopefully) well positioned to complete the project on time (despite some delays). The repository I am using for this project is currently available at: https://github.com/chriskrenz/505-project.

References

1 "SERVICES," damp-lab. Accessed: Sep. 23, 2024. [Online]. Available: https://www.damplab.org/servicesu

2 "PubMed Central: About PMC," PubMed Central (PMC). Accessed: Nov. 06, 2024. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/about/intro/

3 "Bio.Entrez package — Biopython 1.84 documentation." Accessed: Nov. 06, 2024. [Online]. Available: https://biopython.org/docs/latest/api/Bio.Entrez.html#module-Bio.Entrez

4 "NLTK:: Natural Language Toolkit." Accessed: Nov. 06, 2024. [Online]. Available: https://www.nltk.org/

5 "tf–idf," Wikipedia. Jul. 26, 2024. Accessed: Nov. 06, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Tf

6 "Working With Text Data," scikit-learn. Accessed: Nov. 06, 2024. [Online]. Available: https://scikit-learn/ stable/tutorial/text_analytics/working_with_text_data.html