CS505 NLP: Milestone 4 - Project Report

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 am building a machine learning model to take in natural language prompts to predict the user's desired workflow (i.e. sequence of lab services). The DAMP Lab at Boston University offers a manageable number of services from which the model can choose, and the training corpus was acquired through article abstracts on PubMed (for example, references to 'gene editing' or 'modifying sequences' may tend to accompany gene editing services like Gibson Assembly or Modular Cloning). I succeeded in creating a logistic regression classifier using TF-IDF (Term Frequency-Inverse Document Frequency) and One-vs-Rest classification and evaluated the F1-Scores of every class, as well as the model's overall accuracy in sentence-to-service classification. The model achieved a wide range of F1-Scores with an overall accuracy of 58%. In future work I may expand on the model to increase the size/diversity of the training set and address challenges with overlapping classes. With these improvements, the model could be a valuable tool for facilitating the creation and submission of customer orders to synthetic biology laboratories.

1 Goal

My goal is to develop a model that can translate a natural language description of a desired synthetic biology workflow into an actual sequence of specific laboratory services in order to improve the customer ordering experience.

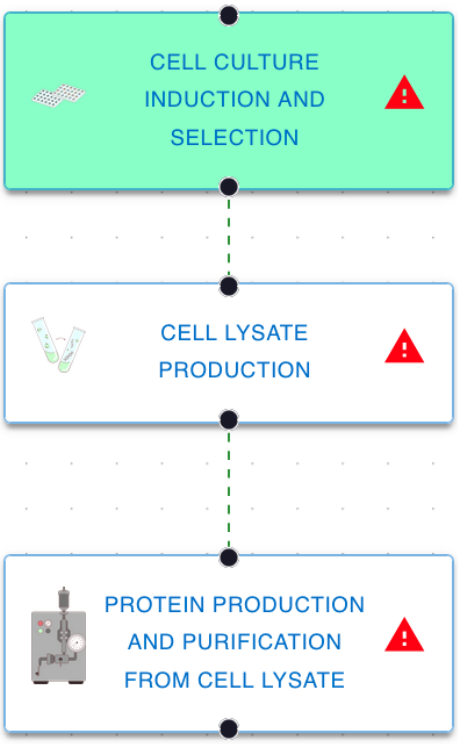

Figure 1: An example workflow in the DAMP Lab ordering system.

2 Background

The DAMP Lab at Boston University offers a manageable number of services (and service bundles) from which the model can choose. Figure 1 shows an example of a sequence of services (i.e. a workflow, or 'bundle') that the DAMP Lab offers. The training corpus was acquired through targeted PubMed searches for these service 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).

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 37 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).

3 Methods

I chose 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 250 articles per keyword, of which, 5,690 had available abstracts. I further decomposed these abstracts into sentences that contain at least one of the keywords/labels, resulting in 3,284 sentences in the training corpus (some abstracts did not include sentences with the keywords in the abstracts). 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 am considering (based on 1-3 sentences, with generally only one label generated from each), services tend to occur in consistent orders. As such, the model will really be predicting a bag of services, rather than a specific sequence; however, at this stage of development, we can reasonably assume that the order in which the services are described is the intended order of the services.

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 **15%**. This comparatively low score is arguably unsurprising, given that model is not trained specifically on synthetic biology data.

4 Model Details

This model uses a Logistic Regression classifier integrated with TF-IDF (Term Frequency-Inverse Document Frequency) feature extraction to per-

form multiclass text classification.

1. TF-IDF Vectorization & Feature Extracts

- IDF: Down-weights terms that appear frequently across multiple documents.
- Transforms corpus into a high-dimensional sparse matrix where each entry reflects the importance of a term in a document relative to the entire corpus.
- $TF - IDF(t, d) = TF(t, d) \times \log\left(\frac{N}{DF(t)}\right)$ where t is a term, d is a document, N is the total number of documents, and DF(t) is the number of documents containing term t .

2. Logistic Regression Classifier

- Models the probability that a given input vector $\mathbf{x}$ belongs to a particular class y.
- For binary classification, modeled as: $P(y = 1|\mathbf{x}) = 1/(1 + e^{-(\mathbf{w}^\top \mathbf{x}+b)})$ where $\mathbf{w}$ is the weight vector, and b is the bias term.
- Utilizing a One-vs-Rest (OvR) strategy, separate binary classifiers are trained for each class against all others.
- The model parameters $\mathbf{w}$ and b are optimized using Maximum Likelihood Estimation (MLE) via Stochastic Gradient Descent (SGD).

3. Training and Evaluation

- Classifier learns to associate TF-IDF features with their corresponding labels by minimizing the loss function, often the Logistic Loss.
- Performance is assessed using Accuracy, Precision, Recall, F1-Score, and Confusion Matrix to ensure balanced and reliable classification across all classes.

4. Mitigating Overfitting

- Applying L2 regularization ($\lambda \sum w_i^2$) penalizes large weights, promoting simpler models that generalize better to unseen data.
- Limiting the number of TF-IDF features and removing high-frequency keywords further reduces the risk of the model overfitting.

5 Experiments & Results

The data were split into 80% for training and 20% for testing. The primary metric I am using is the f1-score, which balances precision and recall. The model achieved an overall accuracy of 58%, with strong performance in classes like 'LB Agar Plate' (f1-score: 0.75). However, certain classes such as 'Protein Purification' exhibited poor performance, likely due to overlapping terminology (e.g. 'peptide' associated with both 'Protein Purification' and 'Gel Electrophoresis'). Other terms, like 'Cell Transoformation' had even lower scores, likely because they are intrinsically more generic concepts. Figure 2 shows a table with the resulting precision, recall, and f1-scores (along with the overall model accuracy) for the first several classes, and Figure 3 shows the confusion matrix corresponding to these classes.

As a specific example, feeding the model the following phrases:

"high separating power immunoblotting synthetic membrane detection peptide"

...correctly yields the label of 'Gel Electrophoresis'. All of these concepts relate to gel electrophoresis (e.g. 'separating power' is the ability of the gel matrix to separate molecules based on size), making this the correct classification.

Finally, the fully trained model can then be used to submit multiple sentences/phrases, each of which will be converted to a corresponding service, collectively forming a workflow.

6 Conclusion

This model effectively translates natural language descriptions into specific synthetic biology services with an accuracy of 58%. It successfully identifies key lab services, but has challenges with overlapping or vague categories. As I continue to develop this in the future, the focus will be on expanding the training dataset, refining keyword mappings, and building on the model to accept multiple sentences and return a sequence of services (i.e. a workflow). With these improvements, the model could become a more reliable tool for facilitating efficient and accurate workflow construction in biological research and manufacturing.

The code has been submitted with this report and is also accessible on GitHub at https://github.com/chris-krenz/505-project. The

```
=== Evaluation Metrics ===
Accuracy: 0.5827
```

Label	Precision	Recall	F1-Score	Support
Gibson Assembly	0.6410	0.6410	0.6410	39
Colony PCR	0.5490	0.5833	0.5657	48
Plasmid Miniprep	0.5000	0.5000	0.5000	2
Restriction Digestion	0.4167	0.4000	0.4082	50
qPCR Assay	0.5263	0.4878	0.5063	41
Modular Cloning	0.6667	0.6429	0.6545	28
Cell Transformation	0.5000	0.2143	0.3000	14
PCR Reaction	0.5128	0.4444	0.4762	45
Spectrophotometric Assay	0.6383	0.6250	0.6316	48
DNA Extraction	0.5652	0.6842	0.6190	76
Agarose Gel Extraction	0.0000	0.0000	0.0000	1
Ethanol Precipitation	0.7458	0.8148	0.7788	54
Concentrate DNA	0.5000	0.4000	0.4444	5
LB Agar Plate	0.7500	0.7500	0.7500	4
Protein Purification	0.4643	0.4062	0.4333	32
Glycerol Stock	1.0000	0.3333	0.5000	3
Gel Electrophoresis	0.6197	0.6667	0.6423	66

Figure 2: Results for the first 5 labels, showing an overall accuracy of 68% and the specific precision, recall, and f1-scores for each label.

```
Confusion Matrix:
[25  3  1  5  0  3  0  1  0  1  0  0  0  0  0  0  0 ]
[ 1 28  0  4  2  0  1  2  0  3  0  1  0  0  2  0  4 ]
[ 0  0  1  0  0  0  0  0  0  0  0  0  1  0  0  0  0 ]
[ 3  6  0 20  1  3  0  2  0  6  0  0  0  0  1  0  8 ]
[ 0  4  0  2 20  2  0  6  2  4  0  0  0  0  1  0  0 ]
[ 4  0  0  2  1 18  0  0  1  0  0  0  0  0  1  0  1 ]
[ 0  0  0  1  3  0  3  0  3  2  0  0  0  0  1  0  1 ]
[ 1  2  0  5  3  0  0 20  0  9  0  1  0  0  1  0  3 ]
[ 1  0  0  2  2  0  0  1 30  1  0  7  0  0  0  0  4 ]
[ 1  5  0  5  4  0  0  4  2 52  0  0  0  0  2  0  1 ]
[ 0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0 ]
[ 0  0  0  0  0  0  0  1  3  2  0 44  0  1  1  0  2 ]
[ 0  0  0  0  0  0  0  0  0  2  0  0  2  0  1  0  0 ]
[ 0  0  0  0  0  0  0  0  1  0  0  0  0  3  0  0  0 ]
[ 2  1  0  0  0  1  2  0  1  5  0  4  0  0 13  0  3 ]
[ 0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  1  0 ]
[ 1  2  0  2  2  0  0  2  4  4  0  1  1  0  3  0 44 ]
```

Figure 3: Confusion Matrix for the above classes.

data are packaged with the code, and the included README explains how to run the code. (If relying on the data packaged with the code, you essentially can run the benchmarker.py script, with the –remove-keywords option, after pip installing dependencies).

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