Protein Data Analysis – Prediction of Enzyme Classification using Protein Sequence Embeddings

Group 4: Ambika Sundaresan, Breanne Baldino, Cindy Yu, Matteo Pinto, Tahamtan Dokhani

Advisor: Dr. Peter Rose, Director of the Structural Bioinformatics Laboratory at SDSC

Abstract

Biologists work with a multitude of protein sequences represented by strings of letters each denoting an amino acid. The amino acid sequence of these proteins allows us to leverage various machine learning Natural Language Processing (NLP) algorithms aimed to predict enzyme classifications, which are indicative of both protein structure and function. We propose a multi-level classification solution that is designed to predict the respective class of a given enzyme. Our approach consists of predicting the classification of an enzyme by applying NLP to a protein sequence. Our method utilizes BERT (Bidirectional Encoder Representations from Transformers) models to create embeddings, or feature vectors, and a variety of machine learning models to predict the respective class of an enzyme.

Table of Contents

1 Introduction and Question Formulation

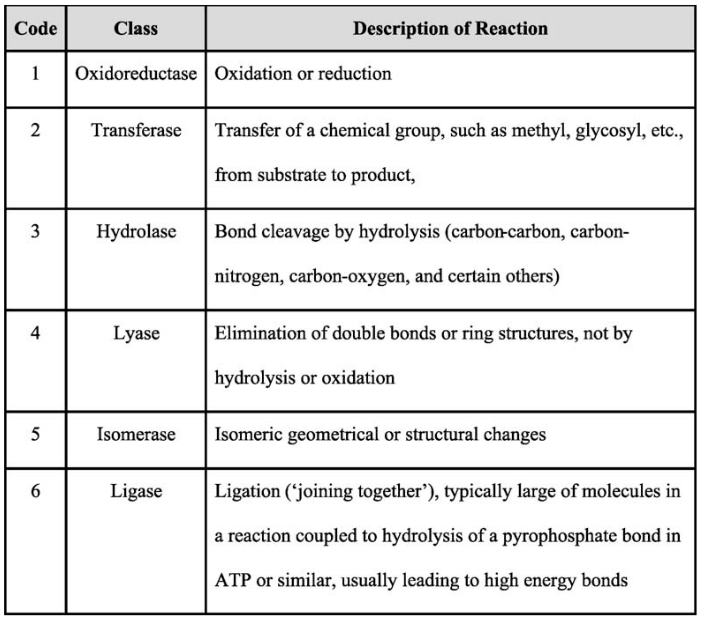
1.1 Protein Data Background

Proteins are essential to life. They are molecules made up of long strings of amino acids, denoted by one of twenty letters of the alphabet, such as:

GCTVEDRCLIGMGAILNGCVIGSGLVAAGLITQ

The sequence of these strings determines its shape, function, or role.

Just as proteins are essential to life, enzymes are essential proteins in the human body that aid in the regulation of biological processes. Enzymes are divided into seven functional classes: oxidoreductases, transferases, hydrolases, lyases, isomerases, ligases, and translocases. It's valuable for scientists to know the classification of an enzyme, as its respective class indicates which type of reaction the enzyme will aid in catalyzing, for example, Ligase: Synthetase.



Code	Class	Description of Reaction
1	Oxidoreductase	Oxidation or reduction
2	Transferase	Transfer of a chemical group, such as methyl, glycosyl, etc., from substrate to product,
3	Hydrolase	Bond cleavage by hydrolysis (carbon-carbon, carbon-nitrogen, carbon-oxygen, and certain others)
4	Lyase	Elimination of double bonds or ring structures, not by hydrolysis or oxidation
5	Isomerase	Isomeric geometrical or structural changes
6	Ligase	Ligation ('joining together'), typically large of molecules in a reaction coupled to hydrolysis of a pyrophosphate bond in ATP or similar, usually leading to high energy bonds

Figure 1: Enzyme Classes and Descriptions

Currently, protein data discovery is outpacing the rate at which enzymes are classified, thus creating a demand for timely and efficient enzyme classification. The methods available for classifying enzymes can be both time and resource intensive. Our goal is to determine enzyme classes of respective protein strings in a time and cost-efficient manner. As mentioned, there are seven different types of enzymes and several additional subclasses per each enzyme class. Based on the data available, we will be performing predictions of enzyme classes for the first six enzyme classes. Because of the hierarchical structure of enzyme classes, our multi-class classification approach will first perform a binary classification to determine if the sequence is an enzyme first, then if the protein is an enzyme, perform a six-class classification to complete the prediction.

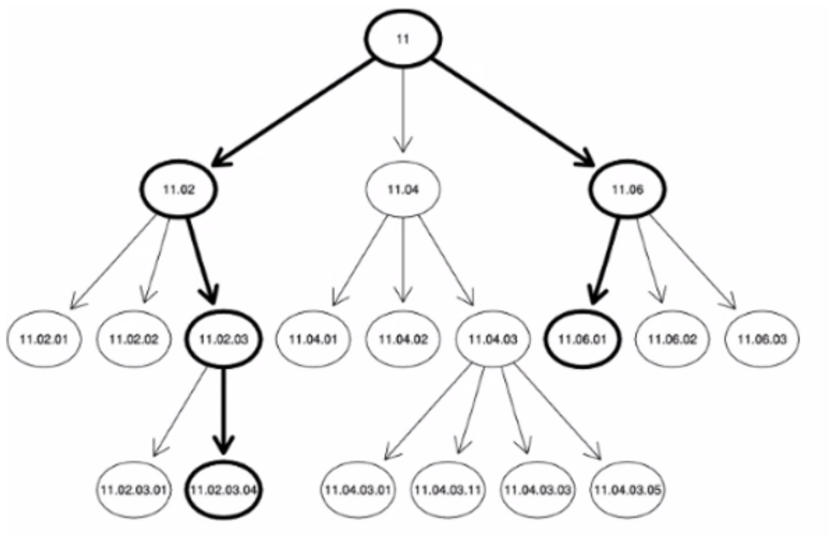


Figure 2: Example of Hierarchical Class Structure

1.2 BERT Models and Natural Language Processing

The amino acid sequence of proteins, denoted by a chain of letters, allows us to leverage various Natural Language Processing (NLP) algorithms to perform enzyme classification predictions – which we pair with Bidirectional Encoder Representations from Transformers (BERT) to predict enzyme classifications in an accessible and affordable way. The pinnacle technique that we have chosen to leverage, BERT, is a semi-supervised model that works by masking certain words in a string, or in our case individual amino acid letters, and aims to predict what those words are by using the context of surrounding words. The output is an embedding which in return, can be fed as input to other machine learning algorithms to perform predictions such as our focus, enzyme classification.

Our goal is to predict protein functional features incorporating pretrained models. The particular BERT models that we are leveraging here are pretrained models known as Tasks Assessing Protein Embeddings (TAPE) and Evolutionary Scale Modeling (ESM-1b). TAPE is a BERT model pre-trained on 31 million protein sequences, developed by researchers at UC Berkeley. ESM-1b is another BERT model trained on 250 million protein sequences by researchers at Facebook AI Research. Their model outperforms all tested single-sequence protein language models across a range of structure prediction tasks. Both research groups used embeddings (weights) as feature vectors derived from their models for downstream Machine Learning models. Feature vectors were then used for prediction tasks such as protein function, protein properties, and structural features.

As such, we will be leveraging the embeddings derived as general feature vectors in our own downstream models. This will be advantageous in that there are no custom feature vectors that need to be designed, and due to the large semi-supervised models, the predictions will be more generalizable than those created from smaller supervised models.

2 Team Roles & Responsibilities

Ambika Sundaresan, Project manager- SME/Visualization developer

Breanne Baldino, Business analyst - Storyteller/coordinator

Cindy Yu, Solution architect - Scalability and operations expert

Matteo Pinto, Solution architect - ML Models and dashboard developer

Tahamtan Dokhani, Data analyst - Visualization and dashboard content

3 Data Acquisition

3.1 Enzyme Classification Data Sources

Our data originated from two main research papers, *ECPred: A tool for the prediction of the enzymatic functions of protein sequences based on the EC Nomenclature (ECPred)* and *DEEPre: Sequence-based enzyme EC number prediction by Deep Learning (DEEPre)*. Both publications detailed methodologies regarding the same enzyme classification we attempted here; the *DEEPre* method utilized a 5-fold cross-validation classification, while the *ECPred* approach leveraged a hierarchical method with a collection of binary classification models. *DEEPre's* dataset contained 22,168 enzyme data and 22,168 non-enzyme data (19 MB) - providing us with a perfect split of enzyme and non-enzyme protein strings alike. While the *ECPred* data contained 249,996 protein sequences with only identifiers for the proteins.

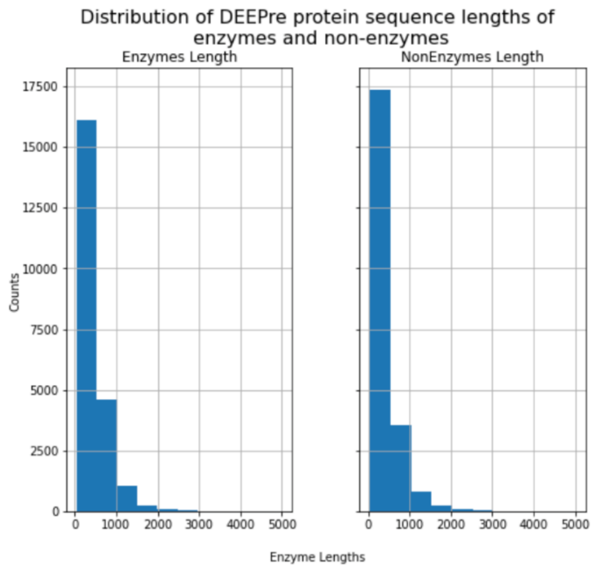
3.2 Data Preprocessing

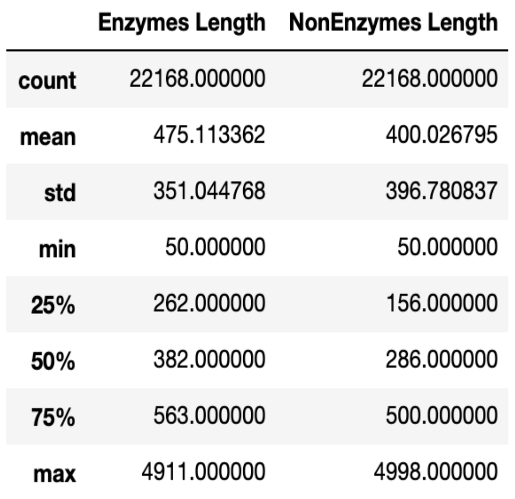
The *DEEPre* Enzyme Data, Non-Enzyme dataset we started with required minimal data cleaning. Preprocessing the data included splitting the text into enzyme label and protein sequence, then inserting these new columns into an empty data frame. For the non-enzymes, we set the labels as "0" for distinction from any of the other classes. We continued to split "label" into enzyme class and enzyme subclass, adding these as additional columns to the data. We wanted to start by processing the data this way so that we had a starting point with accurate labels to classify the enzymes. For data exploration, we experimented with plots to evaluate the most populous enzymes and respective distributions of their class and subclass.

We leveraged two scripts for the *DEEPre* dataset to split the label column into class and subclass. Mirroring the *ECPred* data, we performed a similar split of the label column to organize the data. What differed with this data, as previously mentioned, was that the protein sequences which we required to perform our training were missing. In order to gather the protein sequences for this dataset, we utilized an API call to the fasta file hosted on UNIPROT to obtain each protein id within the data, and extracted a matching protein string. Originally, we called the API sequentially for the 250,000 protein id's to retrieve their respective amino acid chain sequence, but because the data was so large this process was too time intensive. Instead, we modified the pre-processing script to perform the API calls in parallel using multithreading with eight threads. This allowed us to receive around 8X boost in time performance for retrieving these amino acid chains for the dataset.

3.3 Data Exploration

In the *DEEPre* dataset, the total counts of enzymes and non-enzymes are balanced. The amino acid (AA) lengths in each category range from 50 to 4,900, with a median length of 382 for enzymes and 286 for non-enzymes. Less than 5% of the dataset had sequence lengths of greater than 1,000 AA. Considering this, we acknowledged that in the event we face performance or scaling issues, some data exclusions might be necessary. Our first candidates for any data exclusions to mitigate performance issues would be the data of longer lengths (greater than 1000 AA) which would result in exclusion of approximately 5% of the data in *DEEPre.*

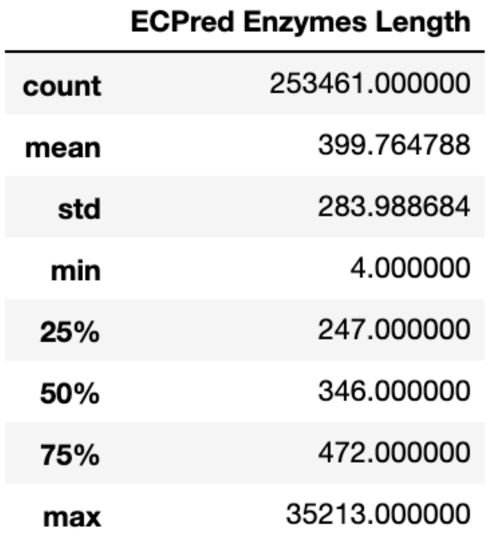




	Enzymes Length	NonEnzymes Length
count	22168.000000	22168.000000
mean	475.113362	400.026795
std	351.044768	396.780837
min	50.000000	50.000000
25%	262.000000	156.000000
50%	382.000000	286.000000
75%	563.000000	500.000000
max	4911.000000	4998.000000

Figure 3: DEEPre Summary Statistics

The *ECPred* dataset (approximately 253,000 AA) included some sequences of longer lengths, up to a maximum of approximately 35,000 AA. Despite these outliers, the median length was 346 AA and the 75th percentile was 472 AA. As in the case of the *DEEPre* dataset, we acknowledged that the AA sequences of a length greater than 1,000 could likely be excluded in our final model.



	ECPred Enzymes Length
count	253461.000000
mean	399.764788
std	283.988684
min	4.000000
25%	247.000000
50%	346.000000
75%	472.000000
max	35213.000000

Figure 4: ECPred Summary Statistics

In addition to the features created in the preprocessing step, our approach for feature engineering and data modeling was to take the sequence data available and load it into our BERT models containerized and deployed on Expanse. The output from our respective models, TAPE and ESM-1b was a 768-dimensional vector and 1,280-dimensional vector, respectively, of pretrained features which were then used to predict the enzyme class and subclass.

The data environment that we developed consisted of flat files that were loaded into the models accordingly, then deployed within a singularity container environment. This allowed for our solution to be more portable and accessible as the flat files were made available through our GitHub repository.

4 Data Preparation and Pipeline

To preprocess the data, we split the protein string text into an enzyme label and an associated protein sequence, then inserted these newly created columns into an empty dataframe. We began processing the data this way so that we could leverage a starting point with accurate labels to classify the enzymes.

Both BERT models provided us with specific constraints that required further data manipulation. Both TAPE and ESM-1b ran into CPU memory limitations due to the models' design of loading the entire input file. Because of both of these constraints, we wrote a script to split the data into smaller partitioned files that we would leverage to obtain our embeddings from both models. For ESM-1b, the data was split into 4 sequences per file, while for TAPE, the data was split into 1,500 sequences per file. In addition to that, ESM-1b had a sequence length limit of 1,024. To mediate that issue, the dataset was reduced and the sequences that were longer than 1,024 AA were removed. The table below details the final dataset utilized per each model:

TAPE		ESM-1b	
Dataset	**Entries #**	**Dataset**	**Entries #**
ECPred	249,996	ECPred	140,584
DeepPre	44,336	DeepPre	26,335

Figure 5: Dataset utilized per each BERT model

The entire dataset was able to be leveraged with TAPE, while exclusions were made for ESM-1b due to sequence length constraints as well as limited compute resources. Specifically, the CPU limitations for ESM-1b required us to further down sample the sequences to 140,584 and 26,335 respectively.

Our data pipeline began with consolidating and preprocessing the enzyme data. After loading fasta files into the singularity container, we utilized the .npz output file for additional downstream tasks. We then performed a binary classification to segment the data into non-enzymes and enzymes. The enzyme data was subsequently used to predict the first level of respective enzyme classes.

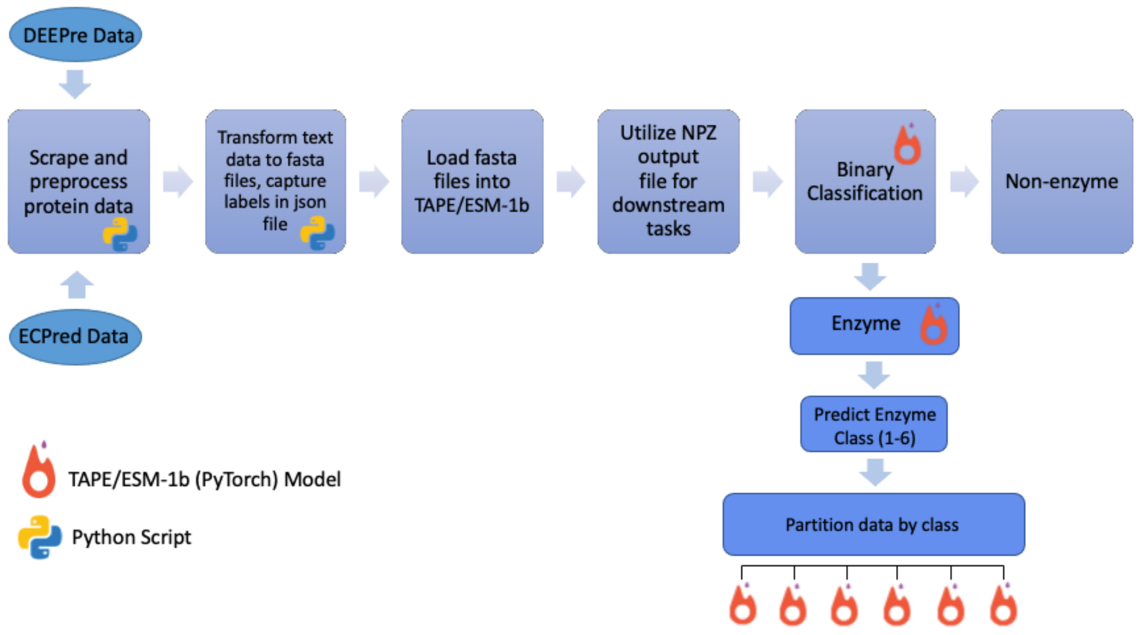

DEEPre Data
Scrape and preprocess protein data
ECPred Data
Transform text data to fasta files, capture labels in json file
Load fasta files into TAPE/ESM-1b
Utilize NPZ output file for downstream tasks
Binary Classification
Non-enzyme
Enzyme
Predict Enzyme Class (1-6)
Partition data by class
TAPE/ESM-1b (PyTorch) Model
Python Script

Figure 6: Data Pipeline utilized to achieve results

5 Analysis Methods

5.1 Exploratory Data Analysis

Exploratory data analysis led us to anticipate and resolve two identified potential data hindrances. First, our protein data did not have inherent features to start, as such, we utilized an n-gram word embedding to generate a simplified feature and used Principal Component Analysis to estimate preliminary model performances and confirm that our data would form expected clusters. Second, our data was not symmetric, in example, each class of an enzyme did not have an equal amount of protein strings contained within it. Through visualizations and descriptive statistics, we identified the amounts of data in each class and subclass and evaluated the distribution.

Ultimately, exploratory data analysis showed us there was a problem with data distribution, each enzyme class had varying amounts of data. As an example, the Isomerases class only contained 15,000 protein strings while the Transferases class contained 90,000 protein strings.

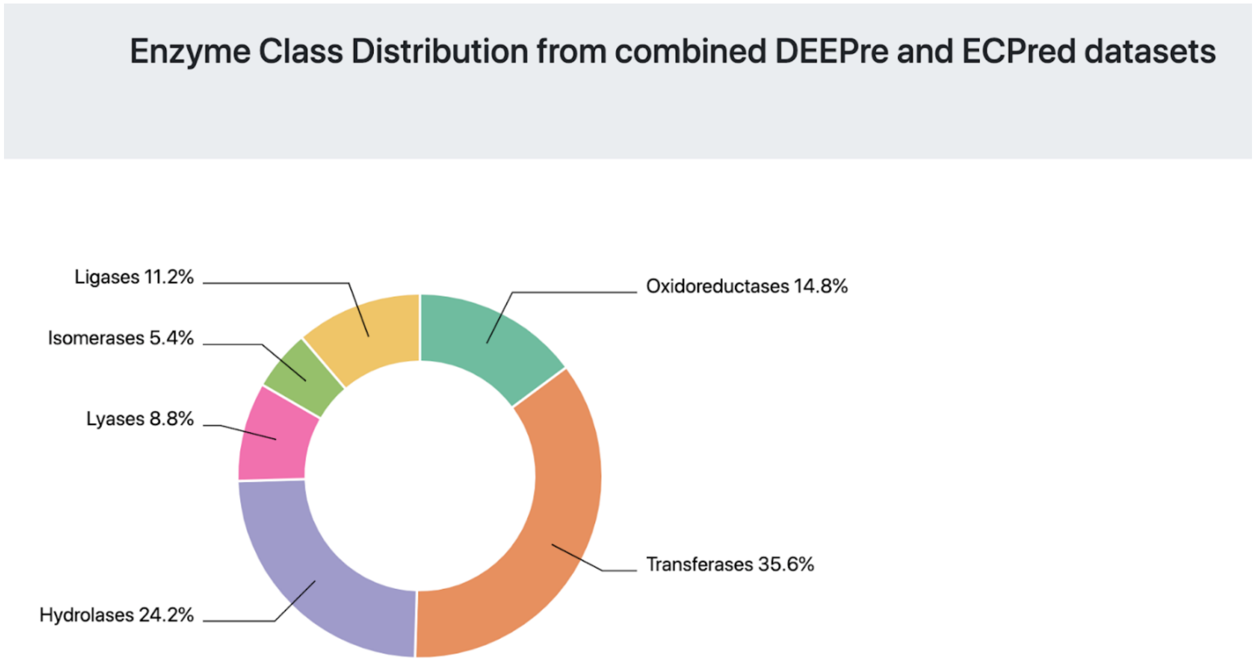


Figure 7: DEEPre and ECPred Enzyme Class Distribution

We explored resolving this imbalance by applying Synthetic Minority Oversampling Technique (SMOTE) to our downstream modeling. SMOTE resolves imbalances in data through synthesizing new examples for minority data classes. This method does this by selecting samples that are close in the feature space, drawing a line between them, and then creating a new sample on a point along that line. Ultimately, our results were not improved with SMOTE and this was not implemented in our end product.

5.2 Model Analysis

In addition to exploratory data analysis, we evaluated our two BERT model approaches TAPE and ESM-1b. To analyze the utilization of ESM-1b, we converted our json data to tuples to make the data compatible with the model. We identified early on that ESM-1b required far fewer libraries to run than TAPE, making it more readily available and accessible. ESM-1b was also touted to have a higher learning rate and learned positional embeddings, potentially making it superior in tasks such as enzyme classification. As noted during our data preparation, ESM-1b possessed a sequence length constraint, making TAPE more versatile in that regard. TAPE was able to utilize GPU to compute embeddings, while ESM-1b was only able to leverage CPU. Both models proved to provide successful word embedding outputs utilized through downstream tasks.

In analyzing how to further improve our model, we attempted to create additional input features for our downstream models. Initially, our model's only input was sequence embeddings and when utilizing a sample of the data, we were only able to obtain about 45 - 65% accuracy for our enzyme class classification. We experimented with feature engineering, such as adding the ratio of how often a character occurs in comparison to the entire length of the amino acid. Since there are 20 various characters that together make up the amino acid chains this gives us about 20 additional features to work with which is a large increase from just the base word embeddings. These new features also provided a more holistic characteristic of the amino acid. After this attempt at feature engineering however, we recognized that additional engineered features could not outweigh the weight that the sequence embeddings held and did not proceed with their utilization.

5.3 Web Application Analysis

Beyond exploratory data analysis, model analysis and tuning, we developed workflows while implementing our Flask application to ascertain we could achieve our desired result.

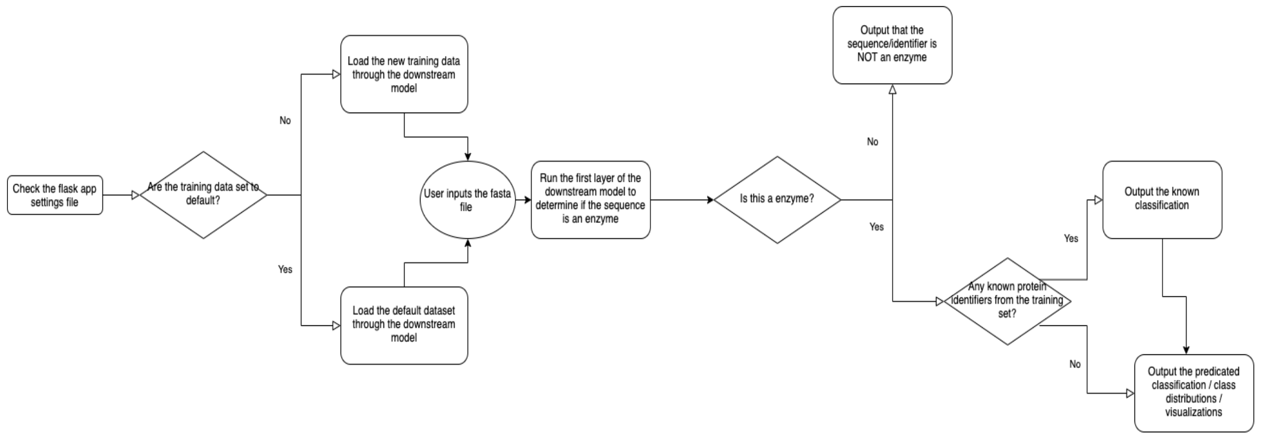

Check the flask app settings file
Are the training data set to default?
No
Yes
Load the new training data through the downstream model
Load the default dataset through the downstream model
User inputs the fasta file
Run the first layer of the downstream model to determine if the sequence is an enzyme
Is this a enzyme?
No
Output that the sequence/identifier is NOT an enzyme
Yes
Any known protein identifiers from the training set?
Yes
Output the known classification
No
Output the predicated classification / class distributions / visualizations

Figure 8: Workflow developed to build Flask Application

Creating a workflow allowed us to adhere to the logic we designed, and thereafter implement it into our application.

Ultimately, analysis methods led us to develop success criteria. We would consider our model successful if our desired end user could successfully obtain embeddings from the singularity container via TAPE or ESM-1b, then bring their embeddings to submit to our Flask Application and obtain predictions derived from the protein sequence embeddings. On top of obtaining embeddings and prediction results, we strived to benchmark these results, leveraging accuracy metrics. We aimed for classification accuracies from our downstream models of above 60%.

6 Findings and Reporting

To validate the performance of our BERT models that produced the protein sequence embeddings, as a first step, we utilized a Principal Component Analysis (PCA) plot to verify that the protein sequence embeddings are separable within the vector space. In addition to using PCA, we also leveraged two other dimensionality reduction algorithms to view the clusters in a two-dimensional perspective, t-SNE and UMAP. We leveraged 2 components and 50 components, respectively, when implementing PCA. While utilizing t-SNE, we leveraged two components on top of both two component and 50 component PCA. Finally, with UMAP we utilized two components. The plots differ in that PCA utilizes the location of features to separate the classes while t-SNE incorporates the probability calculations to predict the classes next to one another. UMAP relies on a weighted graph with weighted edges, representing the likelihood that two points are connected, to segment the classes. t-SNE and UMAP are more successful in clustering the data points because of these techniques.

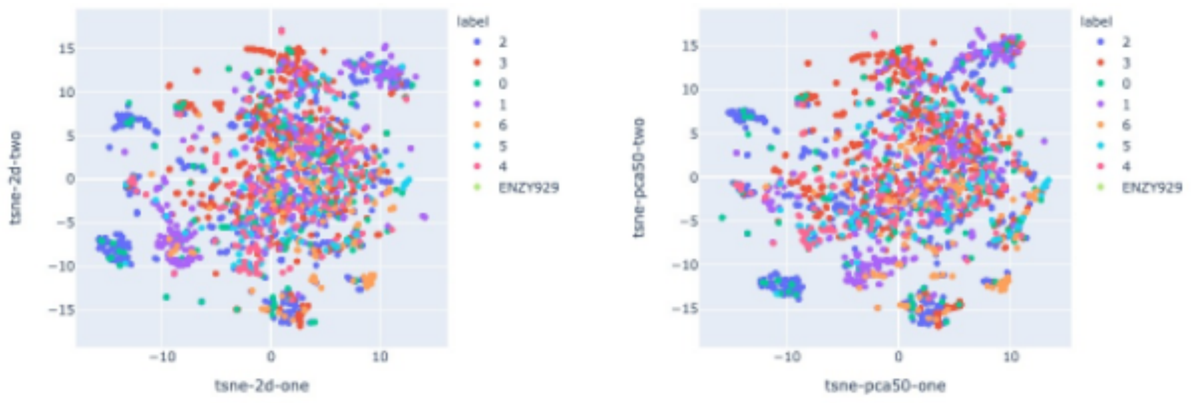


Figure 9: t-SNE plot - DEEPre, TAPE

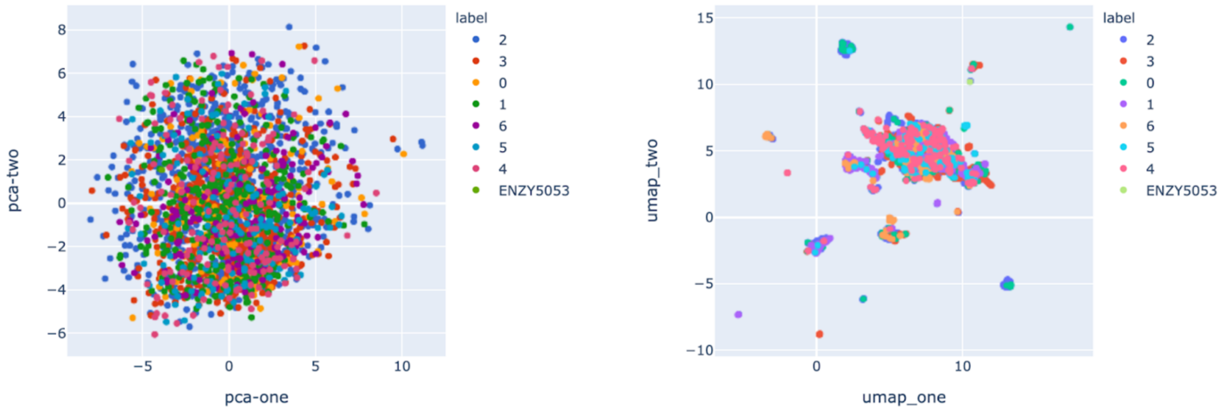


Figure 10: PCA & UMAP plot - DEEPre, TAPE

Subsequent to the BERT models, we used the following machine learning algorithms to perform the multi-class classification: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Multi-Layer Perceptron Classifier (MLP), Random Forest, and Naive Bayes. All these together are strong classifiers that perform predictions in unique ways. They were all hyper parametrized to achieve the most optimal result. Notably, MLP was hyper-parametrized with 300 epochs and 100 layers in the neural network classifier. To obtain hyperparameter optimization throughout all our downstream models, leveraging Python in a Jupyter notebook, we performed a grid search across our test data and selected the best model parameters to use in our final settings.

Model	***Accuracy of Enzyme-NonEnzyme Classes (Binary)***			***Accuracy of Enzyme Classes (6 class classification)***		
	Tape	***ESM-1b***	***Combined TAPE-ESM-1b***	***Tape***	***ESM-1b***	***Combined TAPE-ESM-1b***
K-Nearest Neighbors	*80.8%*	***95.2%***	*96.9%*	*64.8%*	*98.6%*	*96.9%*
Random Forest	*77.6%*	*88.5%*	*96.4%*	*46.3%*	*97.8%*	*96.8%*
SVC	*81.2%*	*91.0%*	*97.6%*	*57.5 %*	*97.2%*	*96.5%*
Naive Bayes	*75.6%*	*85.7%*	*86.3%*	*44.6%*	*62.1%*	*60.7%*
MLP Classifier	***84.1%***	*94.2%*	***98.4%***	***66.1%***	***99.2%***	***98.9%***

Figure 11: Performance of 5,000 embeddings using various downstream models

There are various methods that we used to validate the performance of our downstream models that ingested the BERT model protein sequence embeddings. The first method we leveraged to validate the downstream models was model accuracy, which in our case was the percentage of accurate classifications by our downstream models. We also leveraged a confusion matrix to provide insights on how the model performs regarding classifying a particular enzyme class. The confusion matrix was key in allowing us to understand which classes our particular model was weak in predicting. As well as providing insight on the enzyme classes that need additional iterations in order to improve their accuracy. The values within the confusion matrix are a count of classified enzymes - for each class, the matrix displays how many enzymes were classified correctly and how many were classified incorrectly.

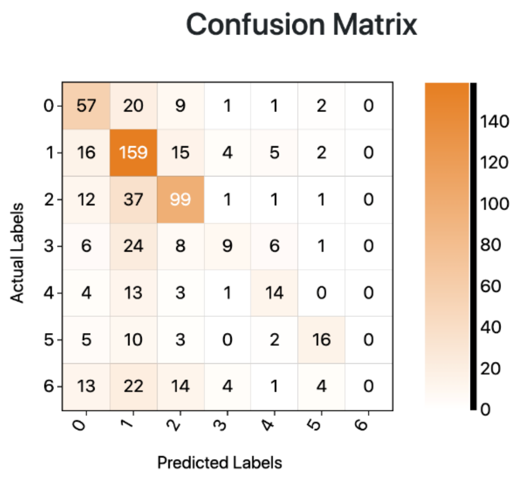


Figure 12: Confusion matrix leveraged to assess model performance on 5,000 TAPE embeddings with the downstream KNN model

In addition to accuracy and the usage of a confusion matrix, we also leveraged F-1 scores through scikit learn to further gauge accuracy. F-1 scores are precision ratings, ranging from 0-1, that measure a model's accuracy through the below formula:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$$

TP = number of true positives

FP = number of false positives

FN = number of false negatives

Figure 13: F-1 Score Formula

Combining both TAPE and ESM-1b embeddings derived the highest accuracy results from our downstream prediction models. We witnessed a significant increase in improvement leveraging both model's embeddings in comparison with each model on their own. Additionally, MLP performed best in classifying enzymes throughout all instances.

Based on the classification results from the downstream models, we identified that the user can gain significant insight on what the most probable enzyme class the sequence belongs to. The downstream model predictions also provided insight on what classes are most distinct and easily identifiable. With the given percentage of confidence for each enzyme class of the sequence, the user can determine how definitive the model was able to identify the class. This also allowed the user to understand the possible confusions between various enzyme classes, as the percentages will indicate how accurate the model was at identifying the correct class. This accuracy data was also valuable in benchmarking how well the embedding inputs perform for each enzyme class identification.

7 Solution Architecture, Performance and Evaluation

In designing the solution architecture, our multi-class classification solution developed into a two-fold pipeline. The first pipeline (BERT models) produced the features from our text or amino acid chain of letters, while the second pipeline (downstream models) performed the predictions.

Our solution architecture incorporated the use of San Diego Supercomputer Center's (SDSC) Expanse. All of the BERT models and feature engineering scripts were migrated into a singularity container in order to seamlessly run on the Expanse infrastructure. This was key not only in training our model, but also in enabling an endpoint for the end user to leverage their own training data by giving them access to this container. The singularity container contained the environment files written in YaML, so that the environment could be recreated, as well as easily duplicated in the event the container is no longer leverageable. The environment files improved the setup time for the environment within the container in addition to keeping every workspace synchronized with the packages and libraries utilized. Two separate singularity containers were created for our use by Martin Kandes at the San Diego Supercomputer Center, one for TAPE and another for ESM-1b to be utilized in Expanse. This allowed for the models to run in their own unique environments, as each model contained separate requirements, specifically the pytorch package version differed for the two models.

Once these features or protein embeddings were produced, they were utilized as input in our downstream models in the second step of our pipeline, the classification process. These downstream models absorbed the features as input and returned the enzyme classification of the respective amino acid. Below is a description of our data product flow that enabled us to achieve our results:

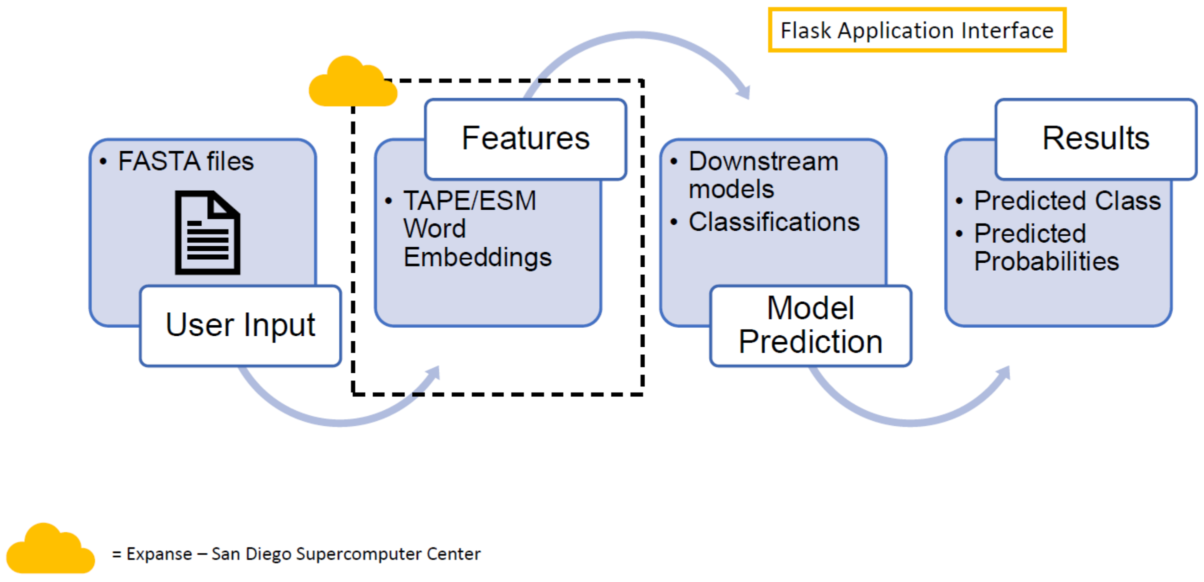


Figure 14: Data Product Flow - Outlines high level steps of Data Solution

In addition to the singularity container hosting the models and feature engineering embeddings, our final component to the solution was our Flask application. We built a web server that allowed the user to access the classification model through a web browser as a better alternative to the command line method. With the Flask application in the architecture, it allowed the server to expose certain endpoints that accessed the models directly. The application had additional endpoints to train the model on certain datasets and utilize different weights to adjust the model for better performance. This web application contained a front-end web user interface which was exposed to our API. Our API allowed users to provide amino acid chain data, and in return our application provided classifications for those chains.

To obtain results, the end user submits their protein data on our application, and of the enzymes included in the user's dataset, our downstream models predict the respective enzyme classification and provide probability confidence per class prediction. To start, the user runs a batch job on Expanse to convert a fasta file to protein sequence embeddings captured in an .npz file. If the user does not have an account, they are directed to access XSEDE User Portal. The user is provided scripts to run either BERT models, TAPE or ESM-1b. Success will be defined by user accessibility as well as accuracy metrics.

8 Conclusions

In pulling together our end-to-end solution, we discovered findings relating to a range of items consisting of most accurate downstream models, feature engineering, and training data optimization. In sum, MLP proved to be the most accurate downstream model, followed by KNN; TAPE & ESM-1b combined embeddings yielded the best downstream results.

In trialing options to improve our models, we experimented with feature engineering and learned that other than incorporating the word embeddings derived from BERT models, incorporating additional features did not improve the accuracy of our downstream models. We also discovered that the embeddings carried such significant weight in our model, that incorporating any additional feature engineering had little to no impact on the results.

As a last step to improve our model, we wrote a script to merge our protein sequence embeddings from both BERT models, TAPE & ESM-1b. Merging our embeddings significantly increased our accuracy results - improving our scores from the 60-80th percentile to 90th percentile. Though this step required additional steps to obtain results, it ultimately proved to be the most successful and promising for future trials by end users.

As a component to communicate results, we incorporated an option to download enzyme classification predictions and class probabilities results as a csv. This provided the user with tangible results relating to their protein data. In addition to this download option, we relied on visualizations to capture a detailed picture of the data of the enzymes contained within their data. Visualizations allowed us to communicate details about the training data we worked with to develop this model as well as key data points for a user's predictions. With this in mind, we developed a dashboard that demonstrates the distribution of the predicted enzyme classes.

Instead of focusing on a narrow picture of the data, our goal was to broaden the lens and provide the user with as much data findings as possible. We also aimed to present insights about our training data within our application. This included PCA, t-SNE, UMAP and K-Nearest Neighbors. These four modalities were leveraged to provide the end user with background relating to enzymes found within the training data. Ultimately, we chose to present these results and related data points through discovery interviews with our target end user and advisor. Following the direction of a domain knowledge expert allowed us to focus on presenting what results mattered as they pertained to enzyme classification.

9 Appendix

a. DSE MAS Knowledge Applied to the Project

Knowledge applied to this project from DSE ranged from PCA - learned in the first quarter of our first year, to machine learning course - learned in the third quarter of our first year to data visualizations learned in the winter quarter of our final year. All of these components worked together to allow us to achieve our end result.

b. Link to the Library Archive for Reproducibility

Baldino, Breanne; Dohkani, Tahamtan; Pinto, Matteo; Sundaresan, Ambika; Yu, Cindy; Rose, Peter (2021). Prediction of Enzyme Classification using Protein Sequence Embeddings. In Data Science & Engineering Master of Advanced Study (DSE MAS) Capstone Projects. UC San Diego Library Digital Collections. https://doi.org/10.6075/J0736QSX

c. References

"GitHub Repository of TAPE." *GitHub*, github.com/songlab-cal/tape.

Facebookresearch. "GitHub Repository of ESM-1b." *GitHub*, github.com/facebookresearch/esm.

Dalkiran, Alperen, et al. "ECPred: a Tool for the Prediction of the Enzymatic Functions of Protein Sequences Based on the EC Nomenclature." *BMC Bioinformatics*, BioMed Central, 21 Sept. 2018, bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2368-y.

Li, Yu, et al. "DEEPre: Sequence-Based Enzyme EC Number Prediction by Deep Learning." *OUP Academic*, Oxford University Press, 23 Oct. 2017, academic.oup.com/bioinformatics/article/34/5/760/4562505.