(2023) 1–5

Advanced Learning for Text and Graph Data (ALTEGRAD) Cellular Component Ontology Prediction

Zakaria ECHCHAIR[b], Marouane NAJID[b]

[a]*Zakaria.ECHCHAIR@polytechnique.edu*
[b]*Marouane.NAJID@polytechnique.edu*

Abstract

In this study, a solution to the Cellular Component Ontology Prediction challenge was proposed. The proposed approach consisted of three parts: utilizing a graph neural network on a graph created from the dataset, using a pre-trained protein language model solely on the sequence, and a combination of both. The approach effectively captured the structural information of proteins and improved classification performance. The best results were obtained by using data from the pre-trained model and incorporating it into the CatBoost model. This study demonstrated the effectiveness of using a combination of different techniques and models for improved protein classification performance.

key words:
Graph, protein, Structure

1. Introduction

In this study, we propose to utilize machine learning and artificial intelligence techniques to address a classification problem in bioinformatics. Specifically, our focus is on proteins, which are vital biomolecules that play a variety of roles within living organisms, including chemical reactions and structural support. The objective of this project is to classify 6,111 protein sequences and their corresponding graph structures into 18 different classes based on their location and function using the Cellular Component ontology.

To achieve this, we employ a two-fold approach that combines graph analysis and natural language processing techniques to gain insight into the protein sequences and structures.

The performance of our models will be evaluated using the logarithmic loss measure, which quantifies the negative log-likelihood of the true class labels given the predictions of a probabilistic classifier.

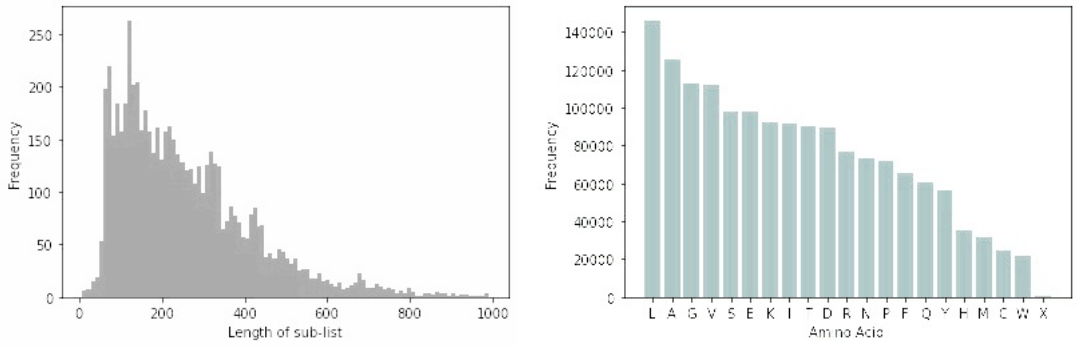
2. Data exploration

In this study, a dataset of 6,111 proteins was utilized, represented as undirected graphs. The protein sequences were obtained from the ”sequences.txt”file, with each line representing the sequence of a single protein. The edges of all proteins were listed in the .[ed]gelist.txt”file, with each line corresponding to an edge represented by the ids of its endpoints. The total number of edges in the dataset was 15,213,222.

Additionally, edge attributes were provided in the .[ed]ge attributes.txt”file, with each line storing the attributes of a single edge. A total of 5 attributes were included, including the distance between the two connected nodes and binary indicators for whether the edge is a distance-based edge, peptide bond edge, k-NN edge, or hydrogen bond edge.

Node attributes were provided in the ”node attributes.txt”file, with each line storing the attributes of a single node. A total of 86 attributes were included, including 3D coordinates of the node, one hot encoding of the amino acid type, hydrogen bond acceptor and donor status, and amino acid features derived from the EXPASY protein scale. The total number of nodes in the 6,111 proteins was 1,572,264.

Graph indicators were provided in the ”graph indicator.txt”file, with the value in each line denoting the graph to which the node with that id belongs. The ”graph labels.txt”file contained the names of all proteins along with the class labels of those proteins that belong to the training set. The proteins for which the class label was not available belonged to the test set, and the goal of this study was to predict the class label of each one of these proteins.



(a) The distribution of sequence lengths is used to specify the input size for transformer-style models that keep the semantic connections between various amino acids.

(b) Distribution of amino acids.

Figura 1: Exploration.

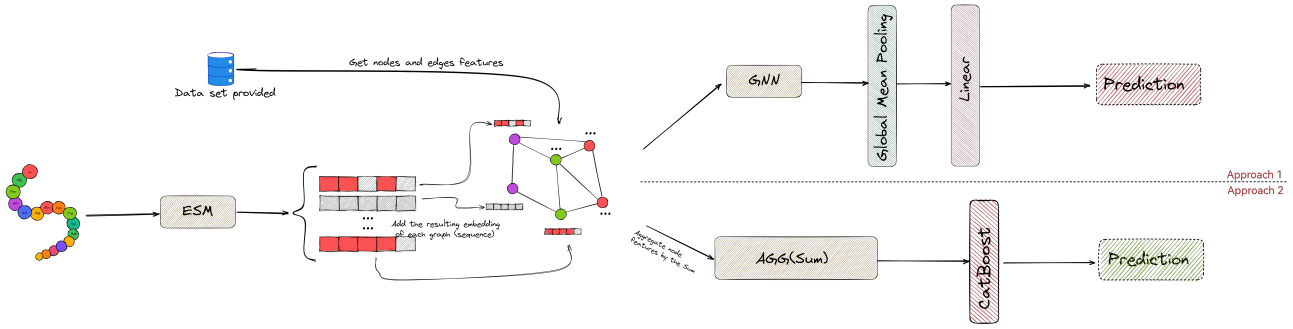


Figura 2: High-level overview of the proposed framework for classifying the different sequences. The framework consists of two main approaches : The first approach involves using a graph neural network (GCN) on the graph created from our dataset, with node features obtained from a pre-trained protein language model (ESM2) applied to the sequence. The second approach involves solely using the pre-trained protein language model (ESM2) on the sequence and then utilizing Catboost for classification.

3. Methodology

The methodology employed in this study to address the problem of protein classification based on graph representations is detailed in this section. The first step taken was a literature review to survey the existing methods and their limitations in this field. Subsequently, three frameworks for protein classification were proposed that build upon the concepts of NLP, graph convolutional networks, and graph attention mechanisms. These frameworks aim to effectively capture the structural information of proteins and improve classification performance.

To evaluate the effectiveness of the proposed frameworks, experiments were conducted on a dataset of 6,111 proteins. The results obtained were then compared to state-of-the-art methods to demonstrate the performance of the proposed frameworks.

3.1. State of the art :

In the study by Elnaggar et al. (1), the use of Language Models (LMs) from Natural Language Processing (NLP) was proposed for protein classification. The team trained several LMs (ProtBert) on a large dataset containing 393 billion amino acids from 2.1 billion protein sequences, using a supercomputer with multiple GPUs. The trained models were able to predict secondary structure, sub-cellular localization, and protein solubility with high accuracy. The team also found that the LMs learned important biophysical properties from the protein sequences. The study demonstrated the success of using large data sets and high-performance computing for training LMs for protein classification, which reduced the gap between models trained on evolutionary information and LMs.

Another study, by Qabel et al. (2), proposed a two-step approach for protein classification based on graph representations. The approach utilized the AlphaFold model, a well-established model in the field, to predict the 3D structure of a protein from its amino acid sequence in the first step. This step aimed to capture the structural information of the protein, which is crucial for accurate classification. In the second step, the sequence was processed using a transformer-based protein language model and a graph neural network was applied to the graph extracted from the structure. The approach aimed to capture the contextual information of the protein, which can also contribute to the accuracy of the classification. The proposed architecture was evaluated on a standard benchmark dataset and it was found to outperform state-of-the-art methods in terms of classification performance.

A typical approach for classifying protein sequences would involve using a combination of different techniques, such as LSTM (3), ProtCNN (4), word2vec (5), fasttext, and TF-IDF for feature extraction, and models such as XGBoost, SVM, and

logistic regression for classification. The LSTM (Long Short-term Memory) is a type of Recurrent Neural Network (RNN) that is particularly well-suited for sequential data such as protein sequences. ProtCNN (Protein Convolutional Neural Network) is a type of convolutional neural network that is effective in recognizing patterns in protein sequences. Word2Vec and Fasttext (6) are word embedding techniques that can be used to convert amino acid sequences into numerical vectors. TF-IDF (Term Frequency-Inverse Document Frequency) is a technique that can be used to weight the importance of amino acids in a protein sequence. Finally, XGBoost, SVM, and logistic regression are models that can be used for classification.

3.2. Approach

In this study, three different approaches were proposed to address the problem of protein classification. The first approach utilized natural language processing (NLP) techniques to process the protein sequences as input, in order to capture the contextual information of the protein. This approach employed techniques such as word embedding and language modeling to analyze the protein sequences. The second approach was based on the graph representation of proteins. This approach aimed to capture the structural information of the protein by representing it as a graph and applying various graph neural network models. The third approach was a combination of the first two, where the sequence information was combined with the graph representation of proteins to exploit both the contextual and structural information of the proteins. This approach aimed to improve the classification performance by leveraging the strengths of both NLP and graph representation techniques.

3.2.1. NLP Approach :

The first approach we proposed in this study is based on natural language processing (NLP) and utilizes only the protein sequences as input. For this approach, we used different embedding methods such as Fasttext and TF-IDF to convert the amino acid sequences into numerical vectors. Additionally, we also used dimensionality reduction techniques such as PCA (Principal Component Analysis) and Auto-Encoder to reduce the dimension of the embeddings. The goal of these techniques is to capture the most important features of the embeddings while reducing the noise.

We then fit these embeddings to various classifiers such as XGBoost, SVM, and logistic regression to classify the proteins. The goal of this approach is to capture the contextual information of the protein by processing the sequence using these embedding methods and then classify the proteins based on their sequence information. The use of dimensionality reduction techniques enhances the performance of the classifiers and reduces the computational cost.

We also employed a LSTM-based approach to classify protein sequences. We used a Long Short-Term Memory (LSTM) network, a type of Recurrent Neural Network (RNN) that is able to capture long-term dependencies in sequential data. We trained the LSTM on our dataset of protein sequences and used it to make predictions about the class of each sequence.

Additionally, we also used a pre-trained embedding model (ProtBERT) to extract embeddings from the protein sequences and fine-tuned it by adding a head of classification to make predictions about the class of each sequence. ProtBERT is a pretrained model on protein sequences using a masked language modeling objective. It is based on the BERT model, which is pretrained on a large corpus of protein sequences in a self-supervised fashion, meaning it was pretrained on raw protein sequences only with no human labeling. This allows it to use a lot of publicly available data and have an automatic process to generate inputs and labels from those protein sequences.

On another hand, we used the ESM (Embedding from Language Models) which is a transformer-based language model that uses an attention mechanism to learn interaction patterns between pairs of amino acids in the input sequence. The ESM-2 model is trained on large amounts of protein sequences data to learn the correlations between different sites in a protein, which can be used to predict missing amino acids and ultimately, the protein's 3D structure. The ESM-2 model's performance improves as the scale of the model increases, as it can learn more complex interactions between amino acids. We utilized the final hidden layer of a pre-trained model and combined it with an ensemble estimator (CatBoost) for classification purposes. Additionally, we attempted to improve the model's performance by fine-tuning it with an additional classification head.

3.2.2. Graph Approach :

In this study, the second approach proposed was based on the graph representation of proteins. This approach aimed to capture the structural information of the protein by representing it as a graph and applying various graph neural network models. In this approach, node attributes and edge attributes were used to classify proteins. Graph Attention Networks (GATs), a specific type of graph neural network that can handle graph-structured data by using self-attention mechanism, were employed. Additionally, other graph neural network models such as Graph Convolutional Networks (GCN), general Graph Neural Networks (GNN), and GraphSAGE were also used. However, some implementation complications with the neighbor data loader were encountered with the GraphSAGE model, which may have affected the results obtained from this model.

These models are well-suited for this task as they are designed to handle graph-structured data and can capture the structural information of the protein. The goal of this approach was to capture the structural information of the protein by representing it as a graph, and apply these models to classify the proteins based on the structural information. The input to these models was the information from both nodes and edges attributes, with the aim of improving the performance by leveraging the graph structure information of the proteins.

3.2.3. Mixture Approach :

The third approach proposed in this study is a combination of the first two approaches, where both protein sequences and graph structure information were used to classify proteins. In this approach, a pre-trained embedding model (ESM) with different architectures and number of parameters was used to ex-

tract embeddings from the protein sequences. Specifically, the last layer of the ESM was extracted as it was believed to contain the most separable representation of the amino acids. These embeddings were then concatenated with the node embeddings and fed to the graph neural network models used in the second approach. Additionally, techniques such as mean pooling layer and dropout layers with a probability of 0.2 were employed to improve performance and prevent overfitting. Furthermore, the GCN model was chosen to have 2 layers based on research that suggests 2 layers as the best architecture for GCN and an increase in the number of layers deteriorates the performance. This approach aimed to capture both the sequence and structural information of the proteins, and by combining them, an improvement in the performance of the classification was expected.

4. Evaluation

After evaluating different approaches for classifying protein sequences, it was found that using simple embedding methods such as tf-idf and fasttext, and fitting them to traditional classifiers like xgboost, svm, and logistic regression, were not sufficient to achieve good results. Therefore, more advanced approaches were explored such as training a transformer from scratch, using ProtCNN to extract features from the sequences, and utilizing pre-trained embedding models like ESM. It was found that incorporating ESM embeddings as inputs for graph-based approaches, such as GATs, GCN, GNN, and GraphSage, along with node features, improved performance. Additionally, fine-tuning ESM by adding a classification head or using only embeddings as input for an ensemble classifier like CatBoost, also performed well. Overall, it was found that these advanced approaches were more effective for tackling the problem of protein sequence classification.

In the training process, the Adam optimizer with a learning rate of $1x10^{-4}$ was used to minimize the multi-class log loss. The model trained was a combination of the ESM2 model and the GNN model on a specific dataset. Different message passing GNN architectures, including GCN, GAT, and GraphSAGE, were experimented with and it was found that GCN performed the best in the setting for the first approach. Although GCN is known to have multiple layers, only 2 layers were chosen for GCN based on research discussions, as it was found that GCN generally reaches its best performance with only two layers. For the other GNNs, 3 hidden layers were implemented. To prevent overfitting, a dropout rate of 0.2 and a batch size of 32 were applied. The GNN had 2 hidden layers for GCN and 3 hidden layers for the other GNNs with a hidden dimension of 64. Dropout, following by layer and elu activation of the GNN, were also used.

As can be seen in the table 1 below, the best results were achieved by using the ESM 3 billion parameters and the Cat boost algorithm.

4.0.1. Evalution metrics

In evaluating our model, we primarily rely on two metrics: accuracy and logarithmic loss function. The logarithmic loss function, also known as log-likelihood, measures the performance of a probabilistic classifier in predicting the true class labels. It is defined as the negative log-likelihood of the predicted class labels.

Specifically, the multi-class log loss is defined as :

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log\left(p_{ij}\right)$$

4.0.2. Results

Tabla 1: Classification results of the diffrents approaches and models on the data set provided. The best performance in typest in **bold**

Models	ACC Train %	Loss Train	ACC Val %	Loss Val
TF-IDF + SVM	81,89	0,29	0,56	1,32
ProBert	82	0,55	70	1,07
Lstm	58,03	1,53	42	1,98
ESM + GCN	82	0,55	70	1,07
ESM + Catboost	82	0,4	75	0,75

5. Conclusion

In summary, this study aimed to address the problem of protein classification using graph representations. A literature review was conducted to survey existing methods and their limitations in this field. Three frameworks were proposed that utilize concepts from NLP, graph convolutional networks and graph attention mechanisms to effectively capture the structural information of proteins and improve classification performance. The effectiveness of these frameworks was demonstrated through experiments on a dataset of 6,111 proteins, and their results were compared to state-of-the-art methods. Overall, the study showed that utilizing large data sets, high-performance computing, and a combination of different techniques and models can lead to improved protein classification performance.

Referencias

[1] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rihawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, D. Bhowmik, and B. Rost, "Prottrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing," *bioRxiv*, 2020. [Online]. Available: https://www.biorxiv.org/content/early/2020/07/12/2020.07.12.199554

[2] A. Qabel, S. Ennadir, G. Nikolentzos, J. F. Lutzeyer, M. Chatzianastasis, H. Boström, and M. Vazirgiannis, "Structure-aware antibiotic resistance classification using graph neural networks," *bioRxiv*, 2022. [Online]. Available: https://www.biorxiv.org/content/early/2022/10/08/2022.10.06.511103

[3] R. C. Staudemeyer and E. R. Morris, "Understanding lstm – a tutorial into long short-term memory recurrent neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1909.09586

[4] M. L. Bileschi, D. Belanger, D. Bryant, T. Sanderson, B. Carter, D. Sculley, M. A. DePristo, and L. J. Colwell, "Using deep learning to annotate the protein universe," *bioRxiv*, 2019. [Online]. Available: https://www.biorxiv.org/content/early/2019/05/06/626507

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2016. [Online]. Available: https://arxiv.org/abs/1607.04606