Protein Cellular Component Ontology Prediction

Waël Doulazmi*, **Ambroise Odonnat***, **Roman Plaud***

Master MVA, ENS Paris-Saclay
{wael.doulazmi, ambroise.odonnat, roman.plaud}@ens-paris-saclay.fr

Abstract

Machine learning for protein engineering has attracted a lot of attention recently. Proteins are composed of sequences of amino acids and have a specific 3D structure. They can be seen as natural language sequences and graphs which motivates the use of graph models in combination with NLP models to solve bioinformatics tasks with proteins. In this paper, we use the sequence and structure of those proteins and classify them into 18 different classes to predict their Cellular Component Ontology. We compare several graph models, mainly GCNs, GATs and a HGP-SL and rely on the ProBert embeddings of amino acids and proteins. We show promising results, achieving an optimal negative log-likelihood of 0.84. We provide an open-source implementation of our experiments at https://github.com/WaelDLZ/AltegradChallenge2022.

Introduction

Proteins are large biomolecules composed of long chains of amino acids. There are 20 different amino acids commonly found in proteins of living organisms. The chain of amino acids is a polypeptide that can be represented by the protein sequence. Moreover, this chain folds and creates the 3D shape of the proteins. Proteins get their chemical property from this 3D structure. This is the reason why proteins can be represented as graphs and also as natural language sequences.

The goal of our work is to use the sequence and structure of those proteins to classify them into 18 different classes, each representing a characteristic of the location where the protein performs its function obtained from the Cellular Component ontology. To tackle this challenge, we drew inspiration from transfer learning which has been successfully applied in Natural Language Processing (NLP) and combine graph-based and NLP methods. In this paper, we first explain the work we did in terms of feature engineering. We mainly relied on the ProBert embeddings of the amino acids as node features (Elnaggar et al. 2021). Then, we briefly introduce the Graph Neural Networks (GNNs) architectures we used, then we describe our framework before presenting our results.

*These authors contributed equally.

Data and Feature Engineering

Dataset

In this challenge, we are given $6,111$ proteins sequences. Each protein can be represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the nodes $\mathcal{V}$ are the amino acids and two nodes are connected by an edge $e \in \mathcal{E}$ based on the Euclidean distance between the nodes, their order in the sequence and the chemical interactions between them. We display in Table 1 statistics about the provided dataset.

Split	$\#\lvert\mathcal{G}\rvert$	Avg. $\lvert\mathcal{V}\rvert$	Avg. $\lvert\mathcal{E}\rvert$
Train	4888	258	4486
Test	1223	254	4379

Table 1: Dataset statistics

Unbalanced Classification

As we can see in Figure 1, the training set is highly unbalanced with some classes more represented than others. To alleviate this issue, we rely on weighting the loss function so that an instance of one of the minority classes weights has more impact on the loss than an instance of one of the majority classes to alleviate the fact that there are seen less often by the model. In practice, the weights are chosen as the squared root of the inverse proportion of each class in the training set i.e for the class i of cardinal N_i, $w_i = \frac{1}{\sqrt{N_i}}$.

Node Features

The nodes, representing amino acids, come with 86 attributes. The first 3 ones correspond to the 3D coordinates of the acid in the protein. We decided to ignore these, as this information is redundant with the distance-based edges.

Then comes the one-hot encoding of the amino acid type. This information is of high importance, as two different amino acids will have different properties. Since one-hot encoding is inconvenient for machine learning as this is a sparse representation, we use tools from Natural Language Processing (NLP) to embed the amino acids, notably the BERT model(Devlin et al. 2019). We will detail this embedding in the next section.

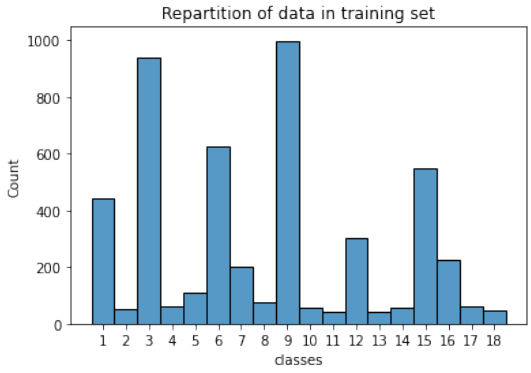


Figure 1: Repartition of data in training set

The 60 last attributes are amino acid features derived from the EXPASY protein scale, representing mainly chemical properties of the proteins. We analyse them through the scope of principal component analysis (PCA) to extract the most relevant information out of those features. To do so, we compile in a dataframe the 60 features of the $1,572,264$ nodes in train and test data. Then, we perform a PCA and look at the part of variance explained by each component. Our results are shown in the Figure 2. We keep the components explaining 80% of the total variance of the features, and thus select 4 components. We plot the projection of each feature on the first two components in the Figure 3. We observe that no feature is significantly more important than the others. Finally, we use the 4 new features (projection on the first 4 components) in our models, in addition with our NLP embeddings for amino acids. We did not notice any performance improvement when using those chemical features, and thus decided to work without them, considering that the amino acids type already conveys enough information.

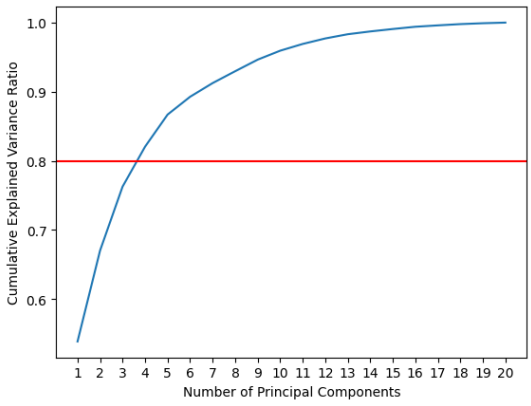


Figure 2: Cumulative explained variance against the number of components retained

Sequence Modelling

The protein sequences can be seen as sentences from the vocabulary made of the 20 amino acids. The baseline proposed

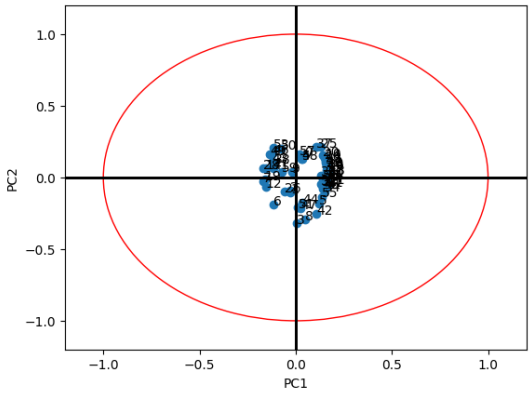


Figure 3: The 60 features projected on the first two components

for the challenge is to use a TF-IDF embeddings for the proteins sequences to perform classification. Pre-trained language models like BERT (Devlin et al. 2019) achieved state-of-the-art results in a wide range of NLP tasks. Inspired by their success, similar models were trained on the protein language for bioinformatics tasks. In our framework, we leverage one of those models, namely ProtBert (Elnaggar et al. 2021), to embed the amino acids types and the protein sequences. ProBert was trained in a self-supervised fashion on the UniRef100 protein corpus (Suzek et al. 2014), a dataset made of 217 millions of proteins. In our work, we rely on the open-source implementation of ProtBert provided by (HuggingFace 2022).

Amino acids

One major strength of BERT is that thanks to its unsupervised training on the masked-language modelling (MLM) task, it produces contextual embeddings for the amino acids. This means that these embeddings convey information not only on the amino acid type, but also about its specific importance in the protein. Thus, using these embeddings of dimension 1024 dimensions as initial nodes features for graph models could help us classifying proteins with a multimodal approach. As shown in the Table 2, it greatly improves the performance of our graph models and enables us to reach great results even with simple graph neural networks.

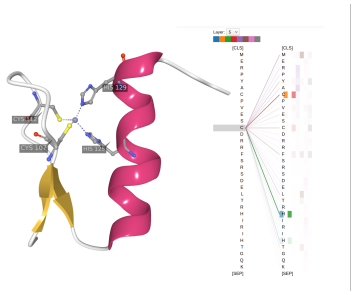
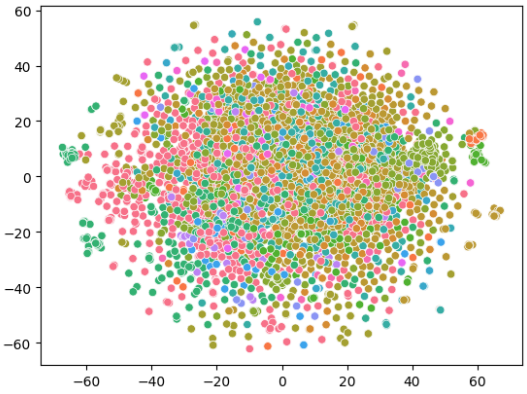
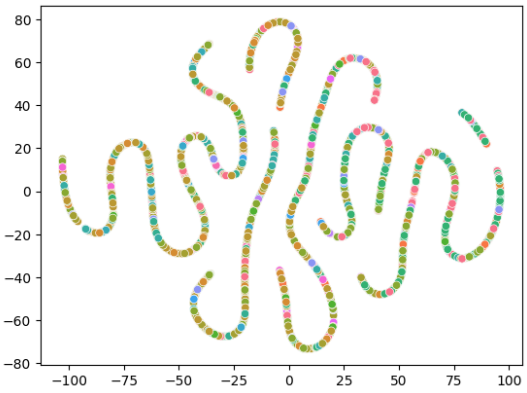


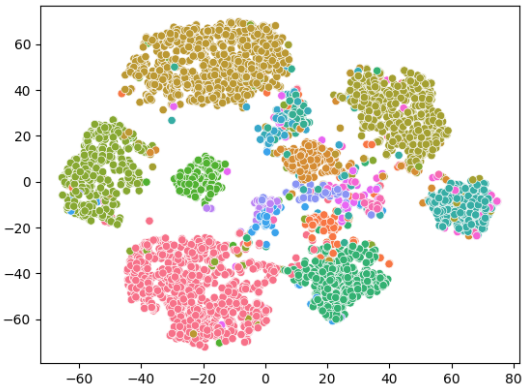
Figure 4: Attention for protein sequences (Elnaggar et al. 2021)



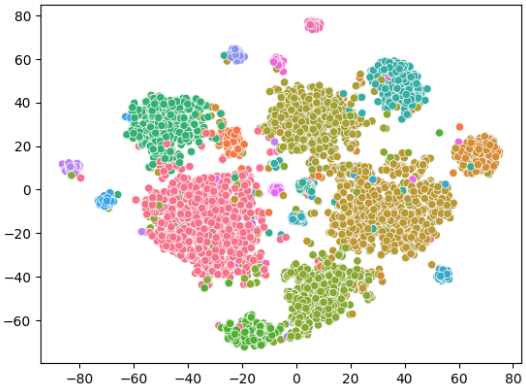
(a) TF-IDF



(b) BERT - No fine-tuning



(c) BERT - Fine-tuned



(d) BERT - Fine-tuned with weighted loss

Figure 5: Visualization with T-SNE of the NLP embeddings of the proteins

Proteins

The last layer of BERT, is a pooling layer that produces an embedding of the whole sequence. We can also use this embedding to perform classification directly on the protein sequences. Using the model without fine-tuning on our data, lead to really poor embeddings as we can see in the Figure 5, where the T-SNE representation of the embeddings shows that the classes are poorly separated. Moreover, these embeddings performed worse than the provided baseline that applies a logistic regression directly on the TF-IDF features of the protein sequences.

Hence, we fine-tuned the model on our classification task, with the help of the `transformers` library (Wolf et al. 2020). We freeze all the layers of the model, except the last BERT layer (for amino acids encoding), the pooler (for protein embeddings) and the classifier. We split our dataset into training and validation, tokenize the sequences using the `BERTTokenizer` that comes with the model, and train it for 500 epochs, following the instructions of the authors of the original Protein BERT paper (Elnaggar et al. 2021). We obtain embeddings of better quality, that separate the classes, without using structural information from the graph representations. The model manages to reach an accuracy of 76.2% in validation, but with poor results in terms of cross-entropy loss (1.46) .

Since the T-SNE embeddings seems to separate accurately the classes (Figure 5), and also give separate clusters on the test set (Figure 6), we consider that they convey salient information on the proteins. If we were interested in the accuracy metric, it seems like using a k-nearest-neighbors classifier directly on the 2D embeddings (BERT + T-SNE) of the sequences would lead to great results (Figure 7). However, we are interested in minimizing the cross-entropy, and such methods might give a 0 probability to the correct class of a sample (take the case of an outlier), making the loss explode. This flaw makes the approach of using shallow learning models on our sequence embeddings non suitable.

This embeddings are still peculiarly salient, and using them in a multi-modal model, combined with structural graph information, leads to interesting results, see the next sections.

Structure Modelling

We consider several types of node features and we compared several ways of connecting nodes by an edge. An edge has 5 attributes, the first one indicating the distance between the two nodes it connects and the last 4 giving its nature: distance-based edge, peptide bond edge, k-NN edge, hydrogen bond edge. We can build several types of graphs by selecting a subset of edges based on their attributes.

Node features We can use the following sets as node features:

- 60 chemical node attributes
- BERT embeddings of amino-acids of dimension 1024
- Combination of both types of features

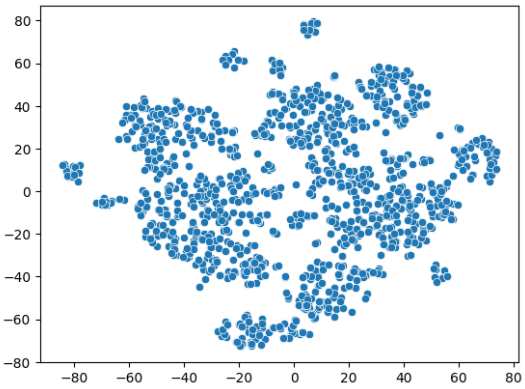


Figure 6: Visualization with T-SNE of the test set embeddings using BERT fine-tuned with weighted loss

Edge features We can use the following sets as edges:

- Use all provided edges
- Use only distance-based edges
- Use only peptide bond edges
- A subset of edges based on their attributes.

In our experiments, we observed that using the chemical node attributes, alone or in combination with the BERT embeddings lead to poor results. This is the reason why we focus on the BERT embeddings as node features.

We also try to combine GNN representations of several graphs representation before applying the classifier, for instance the graph embedding using only distance-based edges and the graph embedding using only peptide bond edges. Although we hoped for improvements as information contained in those representations were different, such combinations were disappointing with a lot of overfitting. The optimal results were obtained using all provided edges and the BERT embeddings of the amino-acids.

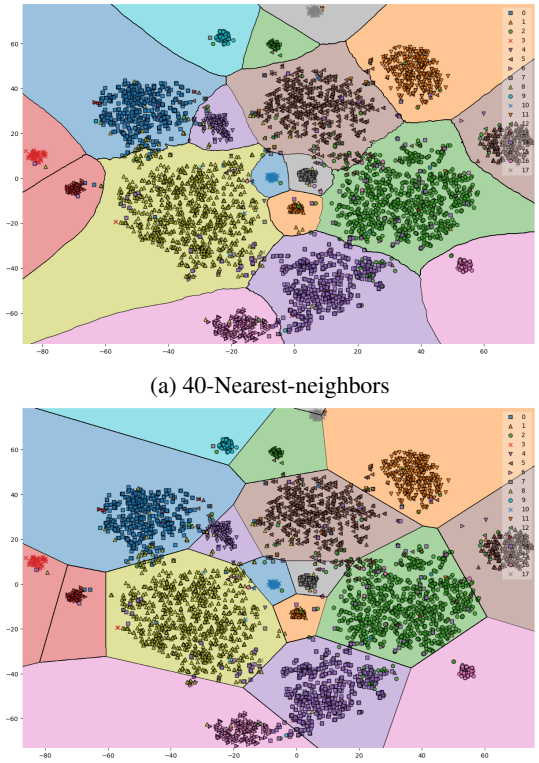
Proposed Method

In this section, we recall the principles of Graph Neural Networks (GNN) before presenting the models we used and giving an overview of our framework.

Graph Neural Network

Various deep learning architectures specifically designed for graphs have been proposed recently. We present GNN by reusing the notations introduced in Thekumparampil et al. (2018). Each node $u \in \mathcal{V}$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is given a node feature $x_u \in \mathbb{R}^k$ with k the node feature dimension. We denote $X = \{x_1, ..., x_{|\mathcal{V}|}\} \in \mathbb{R}^{|\mathcal{V}| \times k}$ the node feature matrix and $\mathbf{A} \in [0,1]^{|\mathcal{V}| \times |\mathcal{V}|}$ the adjacency matrix of $\mathcal{G}$. We consider a model $f(\mathbf{X}, \mathbf{A})$ and denote $\mathbf{H}^{(t)}$ the current hidden states of the layer t where the $i-$th row $\mathbf{H}_i^{(t)}$ is the $n_{hid}^{(t)}$ dimensional hidden state of node i. The message passing layer with respect to a propagation matrix $\mathbf{P} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is:

$$\mathbf{H}^{(t+1)} = \sigma(\mathbf{P}\mathbf{H}^{(t)}\mathbf{W}^{(t)}) \tag{1}$$



(a) 40-Nearest-neighbors

(b) Logistic Regression

Figure 7: Decision boundary of shallow learning classifiers, trained on the 2D protein embeddings.

where σ is an activation function, e.g. a ReLU, and $\mathbf{W}^{(t)} \in \mathbb{R}^{n_{hid}^{(t+1)} \times n_{hid}^{(t)}}$ are trainable weights. For a graph classification task, a $N-$ layer GNN writes

$$f(\mathbf{x}, \mathbf{A}) = \text{softmax}\Big(\bigoplus_{i \in [1, \mathcal{V}]} \hat{\mathbf{H}}_i^{(N)}\Big) \tag{2}$$

with $\hat{\mathbf{H}}^{(N)} = \mathbf{P}\mathbf{H}^{(N)}\mathbf{W}^{(N)}$, P a function of the adjacency matrix A and $\bigoplus$ a permutation-invariant operator such as the sum, averaging or max operator that produces a vector representation of the entire graph of dimension $n_{hid}^{(N)}$. Typically, the propagation matrix $\mathbf{P}$ can be defined as $\mathbf{P} = \mathbf{A} + \mathbf{I}$.

Models

In this work, we implemented and tested three of those GNNs using the Deep Graph Library (DGL) (Wang et al. 2019), namely a Graph Convolutionnal Network (Kipf and Welling 2016), a Graph Attention network (Veličković et al. 2017) and a more recent architecture called Hierarchical Graph Pooling with Structure Learning (HGPSL) (Zhang et al. 2019). The DGL library allows to automatically deal with batches of data, more precisely we do not have to construct the block adjacency matrix as it is done in the `structure_baseline.py` file. In our implementation, we used the sum operator and ReLU non-linearities.

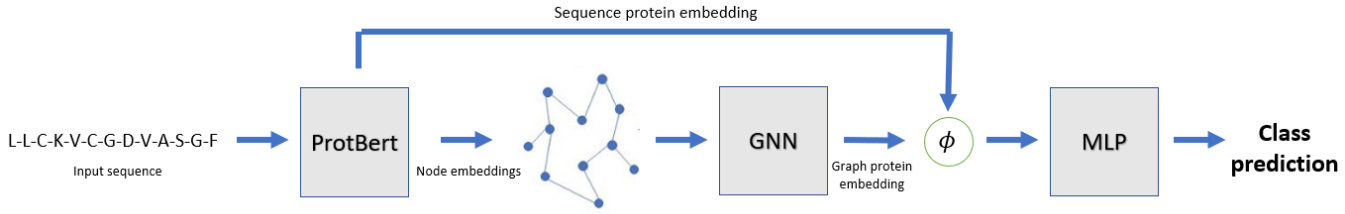


Figure 8: Architecture Overview

Architecture Overview

In our work, we mainly rely on a multimodal framework to combine the NLP sequences and the graph representations of proteins. Indeed, we use NLP embeddings of the amino acids as node features. We can also combine the graph embedding obtained after the GNN and the embedding of the protein sequences. An overview of the proposed method is displayed in the Figure 8. We first use language modelling to generate embeddings of each amino acids and proteins sequences in the dataset. The resulting amino acids embeddings are then used as nodes features in the graph representation of proteins before being fed into a GNN model which outputs the graph embedding of the protein. Finally, we combine the graph embedding and NLP embedding of the proteins before using a Multi-layer Perceptron (MLP) as the classification head.

Numerical Experiments

Implementation Details

We detail here the hyperparameters we tuned during our experiments. Trainings were performed on GPU with 50 epochs, a batch size of 64 and a learning rate at 0.001 with an Adam optimizer. Learning rate is divided by 5 every 5 epochs with a scheduler. We used a split of 90% between training and validation sets and relied on early stopping to avoid overfitting. For the GCN and GAT models, we used 2 convolutionnal layers and a 2-layer MLP as classification head. We reused the default settings of the authors for the HGPSL model (Zhang et al. 2019).

We observed that using the GAT architecture did not yield better results so we focused on the GCN and the HGPSL models. Moreover, we studied the influence of the hidden dimension $n_{hid} \in [128, 256, 512]$ for the convolutionnal layers and compared several types of embeddings combination. We will denote the chosen node features of dimension k as follows:

- G_{all}: using original node attributes, $k = 86$
- G_{BERT}: using BERT embeddings, $k = 1024$

We also denote the embedding of the protein sequences as follows:

- P_{Tfidf}: TF-IDF features of the protein sequences
- P_{BERT}: BERT embeddings of the protein sequences of dimension 1024

Finally, we denote $\text{G}_{\text{BERT}} + \text{P}_{\text{BERT}}$ when we use the graph embedding obtained with BERT embeddings as node features in combination with the BERT protein embedding.

Results

In this section, we show the results obtained on the multi-class classification task and compare the several architectures and training strategies we used. The performance of the models is assessed with the logarithmic loss measure. This metric is defined as the negative log-likelihood of the true class labels given a probabilistic classifier's predictions. Specifically, the multi-class log loss is defined as:

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

where N is the number of samples (i.e. number of proteins), C is the number of classes (i.e. the 18 categories), y_{ij} is 1 if the sample i belongs to class j and 0 otherwise, and p_{ij} is the predicted probability that the sample i belongs to class j.

In the Table 2, we display the loss values of our models. The provided baselines are obtained with a logistic regression (LogReg) on the TF-IDF features of the protein sequences and with a simple GNN with message-passing layers on the 86 original node features. We obtained the lowest losses with the GCN model and BERT embeddings as node features. What is striking is that the HGPSL model does not perform well for our dataset and also that a simple logistic regression on BERT protein embeddings already outperforms the provided TF-IDF baseline.

Model	Embeddings	Hidden dimension	$\mathcal{L}$
LogReg	P_{Tfidf}	$\star$	1.67
LogReg	$\mathbf{P_{BERT}}$	$\star$	**1.161**
GCN	G_{all}	64	1.862
GCN	G_{BERT}	128	0.863
GCN	G_{BERT}	256	0.867
GCN	$\mathbf{G_{BERT}}$	**512**	**0.840**
HGPSL	$\mathbf{G_{BERT}}$	**128**	**1.143**
HGPSL	$\text{G}_{\text{BERT}} + \text{P}_{\text{BERT}}$	128	1.194

Table 2: Loss value on test set for the classification task

Discussion

In conclusion, our work has shown that simple graph neural networks coupled with salient features outperformed more complex models. We relied a lot on large language models embeddings, as they provide powerful tools for natural language processing, even on the protein language. A line of research would be to explore other language models like T5 (Raffel et al. 2020) or XLNet (Yang et al. 2019) that were used in (Elnaggar et al. 2021) and showed promising results.

Our graph models yielded encouraging results, and the text-based approach also proved to be satisfying. Given these positive results, we had high expectations for an approach that combines both. However, we had mixed results with this method. Despite this, we believe that with further investigation and refinement, the multi-modal approach has the potential to go far and achieve great results.

With more hindsight, we realize that we could have explored more advanced tools from graph analysis such as k-core decomposition and graph kernels. These techniques could have enabled a deeper understanding of the underlying structure of the graph data.

References

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Elnaggar, A.; Heinzinger, M.; Dallago, C.; Rehawi, G.; Yu, W.; Jones, L.; Gibbs, T.; Feher, T.; Angerer, C.; Steinegger, M.; Bhowmik, D.; and Rost, B. 2021. ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1.

HuggingFace. 2022. HuggingFace Open-source Probert model. https://huggingface.co/Rostlab/prot_bert.

Kipf, T. N.; and Welling, M. 2016. Semi-Supervised Classification with Graph Convolutional Networks.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Suzek, B. E.; Wang, Y.; Huang, H.; McGarvey, P. B.; Wu, C. H.; and the UniProt Consortium. 2014. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6): 926–932.

Thekumparampil, K. K.; Wang, C.; Oh, S.; and Li, L.-J. 2018. Attention-based Graph Neural Network for Semi-supervised Learning.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2017. Graph Attention Networks.

Wang, M.; Yu, L.; Zheng, D.; Gan, Q.; Gai, Y.; Ye, Z.; Li, M.; Zhou, J.; Huang, Q.; Ma, C.; Huang, Z.; Guo, Q.; Zhang, H.; Lin, H.; Zhao, J.; Li, J.; Smola, A. J.; and Zhang, Z. 2019. Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs. *CoRR*, abs/1909.01315.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zhang, Z.; Bu, J.; Ester, M.; Zhang, J.; Yao, C.; Yu, Z.; and Wang, C. 2019. Hierarchical Graph Pooling with Structure Learning.