MATHÉMATIQUES
VISION
APPRENTISSAGE

Cellular Component Ontology Prediction

JMA team - MVA ALTeGraD course 2022/2023
Jérémie Dentan - Abdellah El Mrini - Meryem Jaaidan

Contents

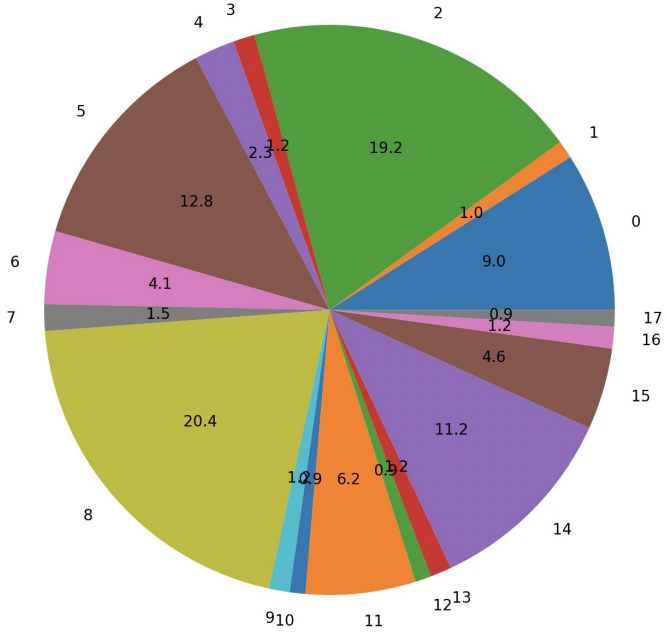
Contents

Introduction

Our task: classify proteins between 18 classes.

Main challenges:

- Unbalanced dataset
- Multimodal data:
  - Node features
  - Edge features
  - Sequences
- Small dataset:
  - Overfitting
  - Lack of information
- No domain-knowledge on the classes

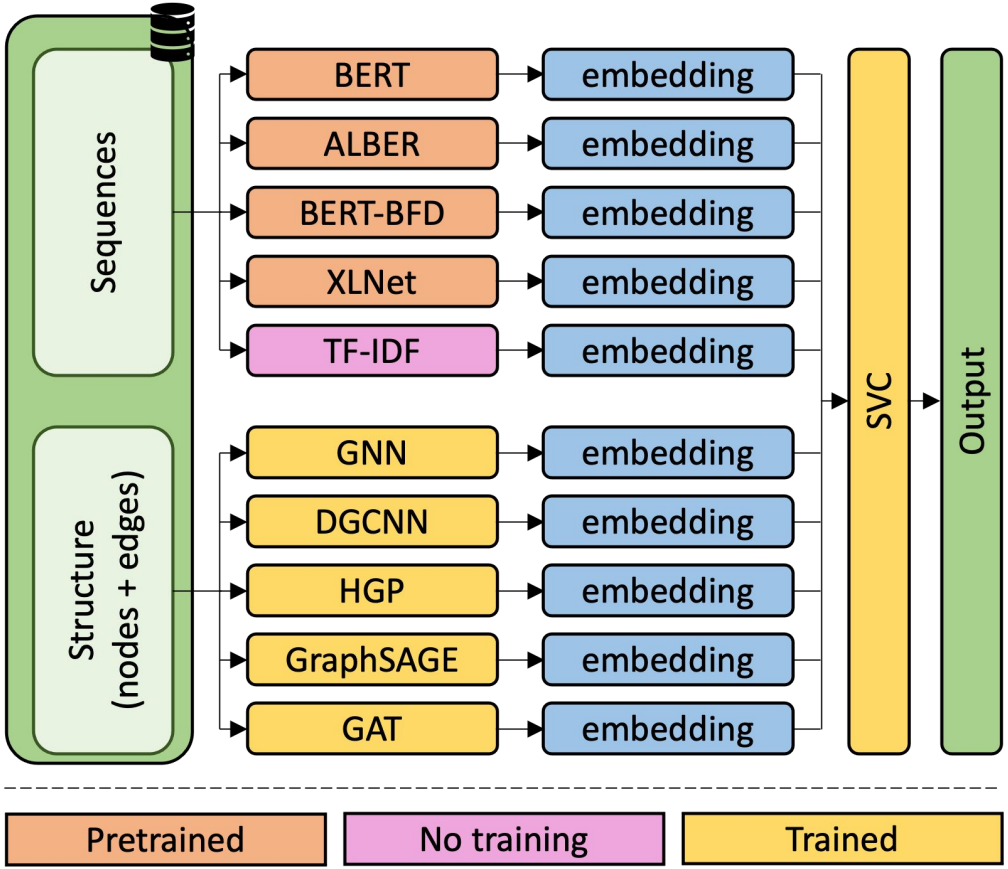


Repartitions of the classes in the train set

Our pipeline

Main idea: embeddings of all proteins.

- Sequence-based embeddings:
  - As text mining with 20 tokens
  - Pretrained LLMs
  - TF-IDF vectorization
- Structure-based embeddings:
  - Message-passing networks
  - Attention-based networks
- A final classifier: SVC



Some computational details

Data and preprocessing:

- 6111 proteins: 4644 train, 244 valid, 1223 test
- Normalization for non-categorical features
- PCA to reduce input size

Computational resources: DIX devices:

- CPU: Intel Xeon W-1290P 3.70GHz 10 cores
- GPU: NVIDIA GeForce RTX 3090 24Go
- But really few disk storage

To avoid overfitting:

- A common validation set for all models
- Early stopping and dropout in models

Use of benchmarks:

- Some benchmarks exist, using the same type of features [4]*
- Hard to reuse since we don’t know the details of our classification task

*cf. the report for the references

Contents

Sequence based embeddings

TF-IDF : Term Frequency - Inverse document frequency

- How meaningful is each chunk of the protein sequence to the sequence ?
- 4-Gram => 151901 features

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Image taken from :
https://mungingdata.wordpress.com/

Sequence based embeddings

Unsupervised protein language models

- Language models on a dictionary of size **20** (number of amino acids).
- Pre-trained on a Masked Language Modeling (MLM) objective.
- Use the embeddings (along with other features) for the downstream classification task. (No extra fine tuning)

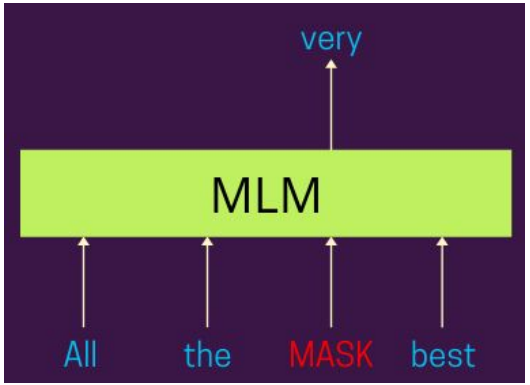
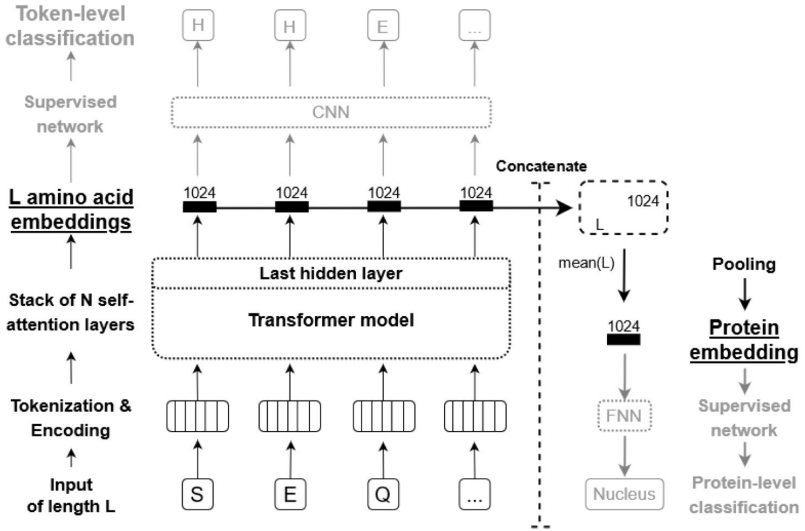


Figure from Towards Data science

Sequence based embeddings



Feature extraction architecture : ProtTrans models

Sequence based embeddings

Transformer-based architectures

- ProtBert (420M) - Based on BERT, a standard NLP model for transfer learning.
- ProtBert-BFD (420M) : Same architecture as ProtBert, but also trained on an additional dataset.
- ProtAlbert (224M) : Hard parameters sharing between attention layers.
- ProtXLNet (409M) : No maximum sequence length required thanks to a memory mechanism.

An issue:

- Sentences in spoken language are 15-30 words long
- Protein sequences are way longer than that **:** Up to 989 in our case.

Contents

Structure based embeddings

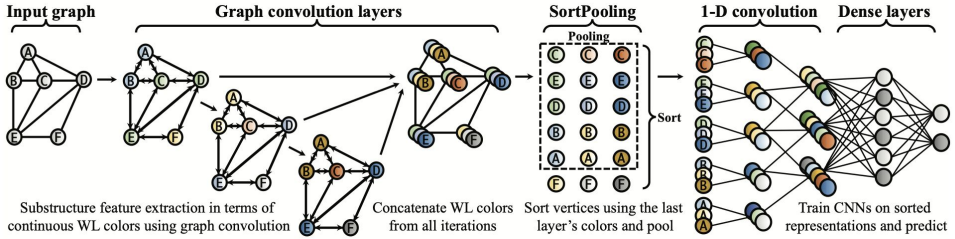
Graph Convolution Network - GCN :

$$H^{k+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{k}W^{k})$$

- Linear operations on the node attributes and its neighbours
- W : weights
- Ã : adjacency matrix
- D : degree matrix with added self loops

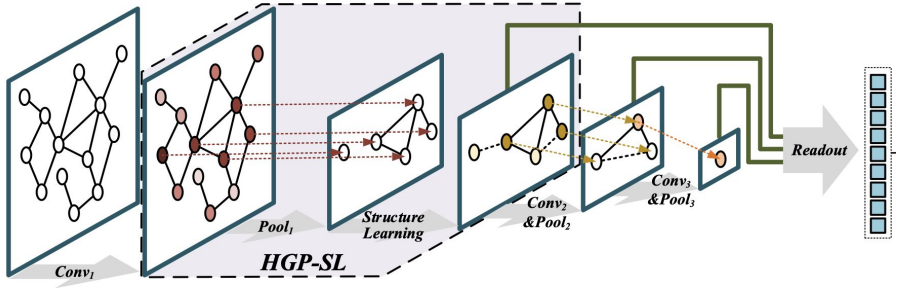
Structure based embeddings

Deep Graph Convolutional Neural Network - DGCNN :



Structure based embeddings

Hierarchical graph pooling with Structure Learning - HGP-SL :

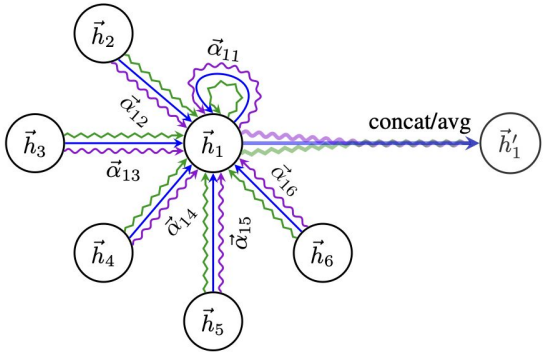


- Graph pooling -> subgraphs - preserve informative nodes
- SL : Learn refined graph structures
- + Preserves graph structure information

Structure based embeddings

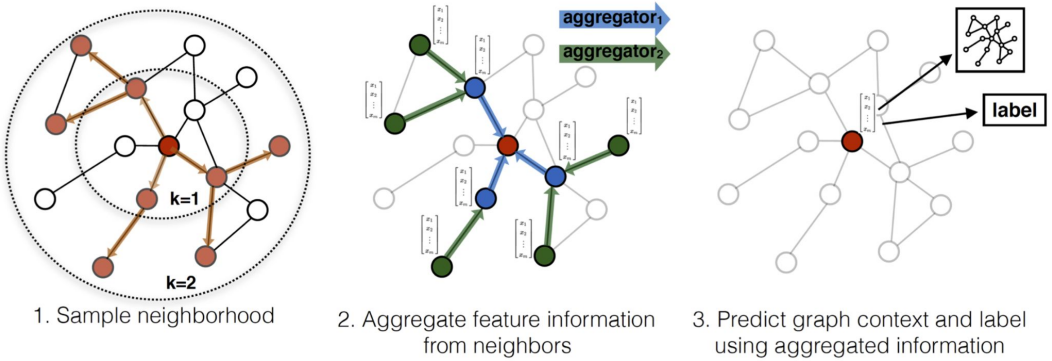
GAT :

- Leverage masked self-attention layers
- Use Graph attention layers
- Different level of importance to different nodes
- Doesn’t depend on the knowledge of the whole graph
- Available with Pytorch Geometric



Structure based embeddings

GraphSAGE :



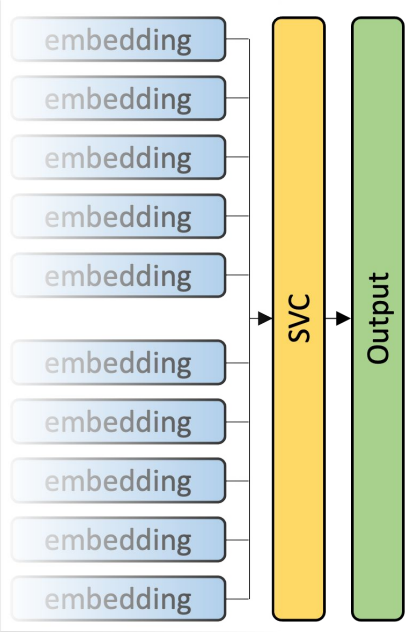
1. Sample neighborhood

2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

Contents

Final classification



Choice of classifier:

- Insight from [4]: SVC behaves very well
- Classifier tested:
  - KNN classifier
  - **SVC -> best performance**
  - Gaussian Process (based on Laplace approximation)
  - Desision tree
  - MLP
  - AdaBoost classifier

Hyperparameter tuning: using our validation set

Feature selection

Why do we need this?

- Some classifier does this natively
- But SVC performances decrease with additional non-informative features

How to select the features?

- State of the art: information theory, correlation with the classification.
- In our case: greedy optimization: at each time, add the embeddings that improve the most.

Emb \ step	1	2	3	4	5	6
BERT	**1.07**	-	-	-	-	-
ALBERT	1.46	NI	NI	NI	NI	NI
BERT-BFD	1.13	**0.97**	-	-	-	-
XLNet	1.49	1.00	**0.93**	-	-	-
TF-IDF	2.43	1.07	0.96	0.93	0.90	NI
GNN	1.70	1.03	0.94	**0.90**	-	-
DGCNN	1.87	1.07	NI	0.92	**0.89**	-
HGP	1.95	NI	NI	NI	NI	NI
GraphSage	1.71	NI	NI	0.93	0.90	NI
GAT	1.98	NI	NI	NI	NI	NI

Performances over the steps of feature selection

Conclusion

Satisfactory performances with reasonable computation time:

- Private score of 0.8467 (16th) and private score of 0.8995 (17th)
- About 1h30 of computation for sequence-based embeddings ; about 20min for structure-based

Many ways of improvement:

- Train the LLMs instead of simply computing embeddings
- Data augmentation: both for underrepresented classes and to avoid overfitting
- Trainable readout
- More advanced method for feature selection and hyperparameter tuning

Final thought: sequence-based embeddings seem to behave better

Thanks for listening !