NLP Homework 1
A.A. 2024/2025

Massimo Romano (2043836)[1], Antonio Lissa Lattanzio (2154208)[2],
[1] romano.2043836@studenti.uniroma1.it [2] lissalattanzio.2154208@studenti.uniroma1.it

1 Introduction

To solve the homework we developed a non LM-based classification based on a **Graph approach** which uses **Label Propagation**. For the LM-based approach we have used different **Transformer Encoder**, testing different types of **Text Input**, different types of **Pooling** and different **Classifiers**.

2 Methodology

2.1 Graph method (Dataset Preprocessing)

We started by conducting some research to identify useful data to include in the dataset. We began by encoding the category and subcategory already present in the provided dataset by using the `sklearn Label Encoder`. Then we proceeded by searching all available properties from Wikidata for each item, constructing a dataframe assigning a value of 0 or 1 depending on whether the property was present in the item's Wikidata information. Next, we analyzed the most common properties and, through visual inspection, selected those that showed significant variance in the number of elements across different classes. See table 1 for an example of selected properties. After this procedure we ended up selecting 26 properties among the total of more than 4300 existing properties. In the next step we downloaded the number of visualizations across 100 different languages, creating a new dataframe where each row corresponds to an item, each column represents a language, and an additional column stores the total number of views, calculated as the sum of all the available pageviews for that item. After various tries we decided to keep only the nodes of visualization in english and the total views. The final set of features we used is a set of 32 features.

2.2 Graph method (Graph Structure)

The final graph structure we propose is a graph made of nodes entity, which contains the labels for the training set, then the nodes are the various column of the dataset with value 0 or 1, for instance the graph will have two nodes for the column "`P17`" that are "`P17_0`" and "`P17_1`", entities with value 0 are connected to the first and entities with value 1 to the second. A particular idea that allowed to increase the accuracy on the validation set was to split the values of the columns "`en`" and "`total_views`" into multiple nodes based on a splitting coefficient that was set to 1000. For instance all entities with a range of visualizations from 0 to 999 in english will be connected to the node "`en_1000`". See figures 1 and 2. The algorithm used for the classification of their labels is the **Label Propagation**: in particular, test set elements are added to the graph without labels and the algorithm propagates labels of neighboring nodes.

2.3 LM-based method (Dataset Preprocessing)

As preprocessing in this case we have added to the original data information about the **English Wikipedia Views** and the **English Wikipedia Summary** of Wikipedia pages. As shown in Table 2 there are the original information provided by the dataset in which we have added the information in Table 3. With all these information we have chosen three different structures (in increasing order based on the information given to the transformer encoder) of the input text (as it is possible to see in Table 4: **ND** (Name-Description), **NDV** (Name-Description-Views) and **NDVS** (Name-Description-Views-Summary).

2.4 LM-based method (Model Structure)

As it is possible to see in Fig. 3, the type of text generated as described in Section 2.3 goes first to a **Tokenizer** (the same of the encoder) that extracts `input_ids` and `attention_mask` using padding ids to reach the `max_length` dimension (or eventually cutting the input sentence to reach that dimension). Then for each token the **Transformer Encoder** extracts in each layer different hidden states, we have taken the `last_hidden_state` of dimension $B \times L \times H$, where B is the **`batch_size`**, L is the sequence length called **`max_length`** and H is the dimension of each hidden vector called **`hidden_size`**. Before sending to the classifier, we need to do **Pooling**: it means to aggregate all these feature vectors in one feature vector that will be sent to the classifier layer (see Section 2.5 for more information). Finally, the aggregated vector is flattened and passed to a **Classifier** layer.

2.5 LM-based method (Pooling Layer)

In this section we will describe the different types of pooling (as is possible to see in Fig. 4). The objective of the pooling is to generate a single output tensor $\mathbf{h}_f$ of dimension F that change depending on the Transformer Encoder type. *CLS* Pooling: take only the token embedding of the `[CLS]` token. *Mean* Pooling: the output is an average of all the token embeddings excluding the padding embeddings. *Max* Pooling: for each components this pooling method takes the max value. *Attention* Pooling: first, it learns how to assign one score s_i for each token embedding using a learnable linear layer. All the scores s_i are then normalized using the **`Softmax`** function obtaining α_i and finally the output

$$\mathbf{h}_f = \sum_j \alpha_j \, \mathbf{h}_j$$

3 Experiments

3.1 Graph Method

In the experimentation with the graph method we tried different combinations of features in the dataset and also with different values of split for the columns "en" and "total_views". We found that putting too many informations in the graph leads to poorer performances and longer inference time for the label propagation. We also tried to change the structure of the graph by creating the properties node only when they have value "1" but this also led to worser performances. We also tried different combinations of the hyperparameters of the LPA. In general we found that the best value for the knn is 10 and 15 for the rbf. You can see some examples in table 6. An experiment that was completely discarded was to find useful words in the descriptions of the items by analyzing the top most used words and adding their presence to the dataframe as binary values but this did not lead to improvements.

3.2 LM-based Method

For the LM-based method we have tried different types of text in input (**ND**, **NDV**, **NDVS**), different Transformer Encoder (**BERT-base**, **DeBERTa Base**) and different pooling type (**CLS Pooling**, **Mean Pooling**, **Max Pooling**, **Attention Pooling**). We have also tried two different types of classifier: **Linear** layer and **MLP** with hidden dimensions **[F, 128, 3]** and activation function **ReLU**. The hyperparameters used are showed in Table 5. To avoid overfitting for the fine tuning, the number of epochs chosen is small. The optimizer is **Adam** and as loss function we used the **Cross Entropy Loss**. During training, the best model was saved on the basis of the lowest loss on the validation set. All the results showed are on the **validation set**.

4 Results

4.1 Graph Method

The best result obtained with the LPA is shown in table 7 with the confusion matrix shown in figure 5. This result is obtained with the KNN kernel with 10 neighbors.

4.2 LM-based Method

First of all keeping fix the Transformer Encoder to **BERT-base** and type of input text to **ND** in order to test the different types of pooling, seeing in Table 8 that the **Attention** pooling is the best in terms of metrics but it will increase the average inference time. Then we have chosen as pooling mechanism the **Attention** to test the classifier: as it is possible to see in Table 9 the MLP is the better choice. The last test was to measure the performance of different Transformer Encoders in combination with different input text type, using pooling **Attention** and **MLP** as classifier stage. In Table 10 it is possible to see that **DeBERTa-base + NDVS** combination obtain the best results (in fact it is the model used to produce the predictions in the test set), that in general adding the views to the input text do not improve performance, but the Wikipedia Summary increase a lot, and finally that the **NDVS** input text type helps the model also to decrease its Average Inference Time. Finally in Fig. 6 is possible to see the confusion matrix of the **DeBERTa-base + NDVS** combination.

Property	Total	C.A.	C.R.	C.E
P17 (Country)	1873	96	364	1413
P214 (VIAF cluster ID)	1637	39	642	956
P856 (Official website)	1312	80	463	769
P213 (ISNI)	1140	11	537	592
P106 (Occupation)	1022	10	491	521
P21 (Sex or gender)	1022	11	491	520

Table 1: Distribution of some labels for selected properties from Wikidata.

Item	Name	Description	Type	Category	Subcategory
`http://www.wikidata.org/entity/Q84`	London	capital and largest city of England and the United Kingdom	entity	geography	geographic location
`http://www.wikidata.org/entity/Q23402`	Musée d'Orsay	art museum in Paris, France	entity	media	production company
`http://www.wikidata.org/entity/Q1772945`	Opera dei Pupi	typical Sicilian marionette theatre	entity	performing arts	theatrical genre

Table 2: Basic Dataset

Views	Summary	Label
3 760 534	London is the capital and largest city of both England and the United Kingdom, with a population of 8,866,180 in 2022. Its wider metropolitan area is the largest in Western Europe...	cultural representative
251 102	The Musée d'Orsay is a museum in Paris, France, housed in a former railway station. It holds mainly French art from 1848 to 1914, including the world's largest collection of Impressionist paintings...	cultural representative
7 322	The Opera dei Pupi is a marionette theatrical representation of Frankish romantic poems traditionally performed in Sicily, Italy. It was inscribed in UNESCO's List of the Oral and Intangible Heritage...	cultural exclusive

Table 3: Wikipedia views, summaries, and cultural labels

Text Type	Example Sentence
`ND`	"*Name: Opera dei Pupi. Description: typical Sicilian marionette theatre*"
`NDV`	"*Name: Opera dei Pupi. Description: typical Sicilian marionette theatre. Views: 7322.0*"
`NDVS`	"*Name: Opera dei Pupi. Description: typical Sicilian marionette theatre. Views: 7322.0. Summary: The Opera dei Pupi is a marionette theatrical representation of Frankish romantic poems traditionally performed in Sicily, Italy. It was inscribed in UNESCO's List of the Oral and Intangible Heritage...*"

Table 4: Input Text for different text types

Hyperparameter	Value
Batch Size	128
Learning Rate	1×10^{-4}
Epochs	5
Max Sequence Length	64

Table 5: Training Hyperparameters

RBF Kernel		KNN Kernel	
Gamma	Accuracy	n_neighbors	Accuracy
10	0.69	**10**	**0.736**
20	0.68	20	0.69
50	0.67	50	0.68

Table 6: Accuracy for different kernel parameters on the validation set

Class	Precision	Recall	F1-Score	Support
0	0.6346	0.8684	0.7333	76
1	0.8246	0.4393	0.5732	107
2	0.777	0.9231	0.8438	117
Accuracy		0.7367		
Macro Avg	0.7454	0.7436	0.7168	300
Weighted Avg	0.7579	0.7367	0.7193	300

Table 7: Classification Report for the graph based method

Pooling Type	Accuracy	Precision	Recall	F1-Score	Inference Time Avg (s)
CLS	0.740	0.7169	0.7238	0.7165	**0.0119**
Mean	0.750	0.7306	0.7369	0.7280	0.0150
Max	0.757	0.7389	0.7450	0.7379	**0.0119**
Attention	**0.763**	**0.7524**	**0.7578**	**0.7464**	0.0352

Table 8: Performance comparison of different pooling types using `text_type : ND` and `language_model_name : bert-base-uncased`

Classifier Type	Accuracy	Precision	Recall	F1-Score	Inference Time Avg (s)
Linear	0.763	0.7524	**0.7578**	0.7464	0.0352
MLP	**0.777**	**0.7582**	0.7558	**0.7559**	**0.0158**

Table 9: Performance comparison of classifier types using `text_type : ND` and `language_model_name : bert-base-uncased` `pooling_type : attention`

Transformer Encoder	Text Type	Accuracy	Precision	Recall	F1-Score	Inference Time Avg (s)
BERT-base	ND	**0.777**	**0.7582**	0.7558	**0.7559**	0.0158
	NDV	0.753	0.7385	0.7413	0.7339	0.0101
	NDVS	0.767	0.7519	**0.7574**	0.7517	**0.0095**
DeBERTa-base	ND	0.767	0.7508	0.7561	0.7503	0.0241
	NDV	0.760	0.7476	0.7517	0.7448	0.0237
	NDVS	**0.780**	**0.7690**	**0.7716**	**0.7643**	**0.0231**

Table 10: Performance comparison by input text type across Transformer Encoders (Pooling: Attention, Classifier: MLP)

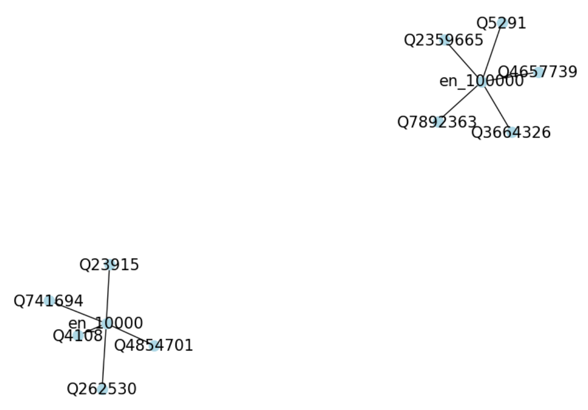

Figure 1: Example of node en splitted in many nodes based on the value, more specifically here there are two nodes en_10000 and en_100000. This image was produced by the graph made with the training set.

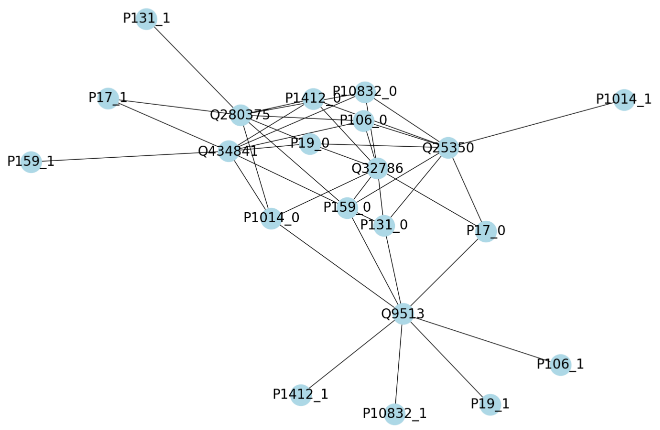

Figure 2: Example of nodes connected to some properties, the name "PXXX_0" means that the item connected to it does not have that property in wikidata. This image was produced by the graph made with the training set.

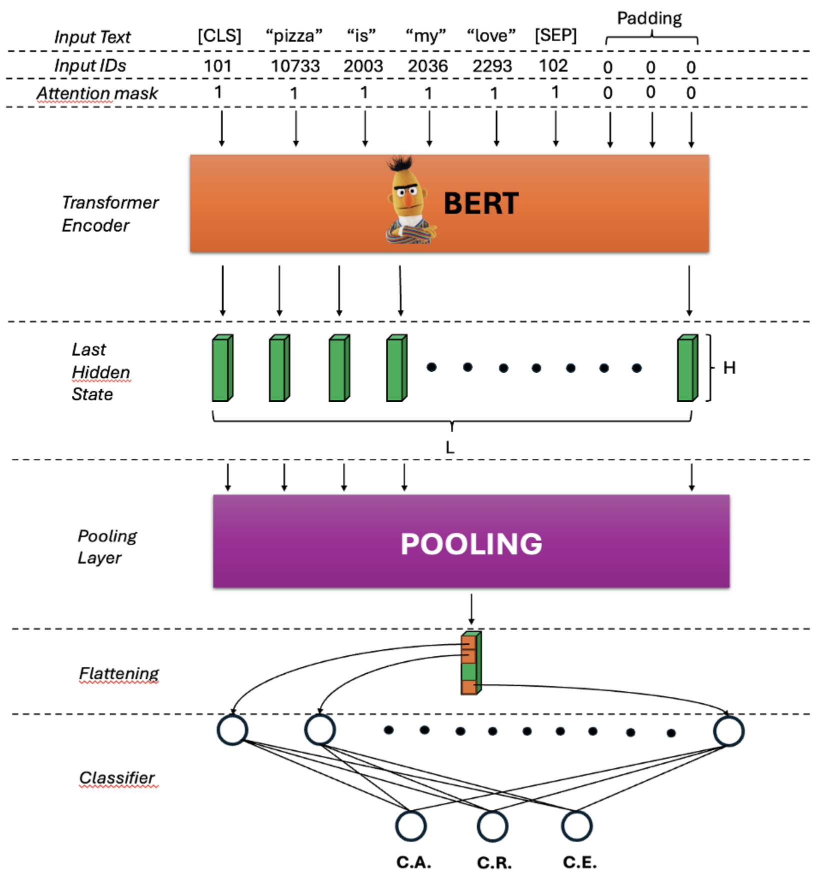

Figure 3: LM-based Method

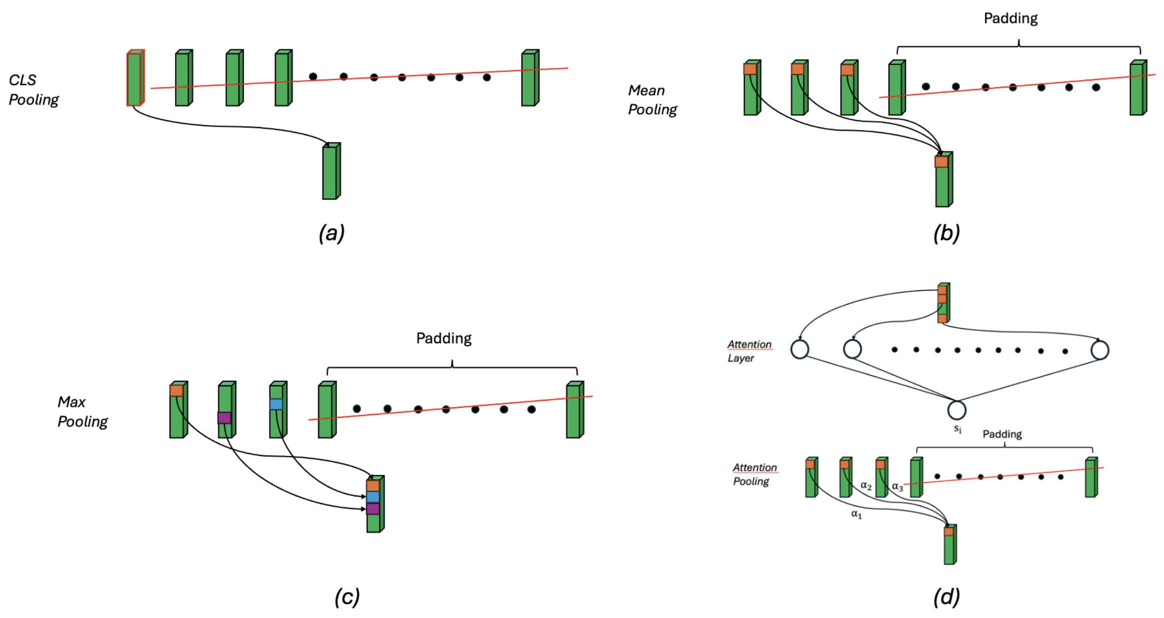

Figure 4: (a) CLS Pooling (b) Mean Pooling (c) Max Pooling (d) Attention Pooling

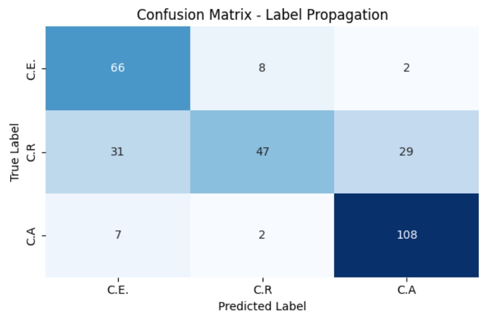

Figure 5: Confusion matrix for the graph based method with LPA and knn = 10

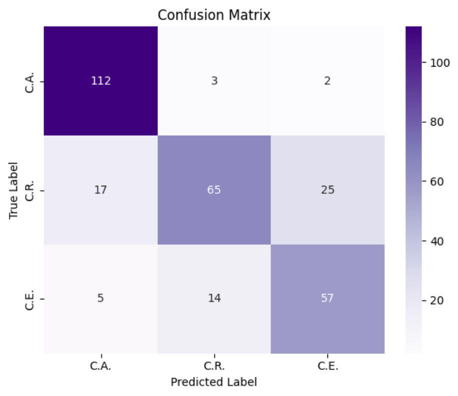

Figure 6: Confusion matrix DeBERTa-base + NDVS + Attention + MLP