Extreme MultiLabel Text Classification
Athens 2021

Calendar

Date	Targets
April-2-2021	Based on LAST-SIX method of Chalkidis paper, build a similar model for Rapterchis-47k dataset
May-5-2021	• Test other Bert version like, Greek Legal Bert by athinaios and MultiLingual Bert • Use Legal Bert and similar ideas on other datasets (EURLEX)

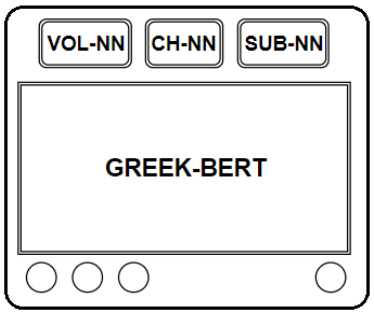
Papers:

- Layer-wise Guided Training for BERT: Learning Incrementally Refined Document Representations (Chalkidis et.al. 12 Oct 2020)
- An Empirical study on LMTC Including Few and Zero Shot Labels (Chalkidis et.al 4 Oct.2020)
- Parameter Efficient Transfer Learning for NLP (Houlsby et.al 2Feb. 2019)

Raptarchis47k:

Raptarchis47k dataset contains greek legislation documents from the period of 1834 until 2015. Its labels follow three level hierarchical structure consisted of Volumes (level-0) Chapters (level-1) and Subjects (level-3). For the Classification process we will use the header of each document and the body/articles which follow the header.

Introduction



In this round of the experiments, we train the models on Raptarchis47k dataset and therefore we need a Greek version of Bert able to understand and handle Greek. Hence, It is used in the Greek Bert which returns as output a dictionary with “logits” and “hidden_states”. The below models try to exploit these different layers and to do so they use ***only*** embeddings from the “hidden_states” output.

Configuration: All the models use the same configuration which means that they have:
MAX_LEN = 400, BATCH_SIZE = 4, MAX_PATIENCE = 3 and LearningRate = 1e-5.

Loss Function: For the loss function we chose pytorch’s CrossEntropyLoss which combines LogSoftmax and NLLLoss (negative log likelihood) in one class. This practically means that we do not apply any activation function on the outputs of the final Neural Networks (VOL-NN, CH-NN, SUB-NN) which are used to predict the volume, chapter and subject of a given instance.

Models:

The models which have been developed are the following:

SimpleBert:

The first approach is a simple Flat structure, in which after the Bert Layer the [CLS] token is used as input to three distinct Neural Networks, the VOL-NN to predict the Volumes (dim. 768xUniqu_Volumes), the second CH-NN for Chapters (dim. 768xUniq_Chapters) and the last one the SUB-NN for the Subjects (dim. 768xUniq_Subjects).

The final loss is the sum of Cross Entropy Losses of these Neural Networks.

Last3_Bert:

The equivalent LAST-SIX architecture from Chalkidis et al. paper for Raptarchis47k dataset. In this approach, the -3 layer (Hidden_States[-3]) is used to predict Volumes, the -2 layer (Hidden_States[-2]) is used for Chapters and the last layer (Hidden_States[-1]) for Subjects.

There are also other variations of this model in the paper (like In-pairs, one-by-one or hybrid), which have not yet been implemented.

Mask_SimpleBert:
In order to interconnect the predictions of the higher layers with those of the lower layers, we use a masking technique so as to prevent the appearances of labels which are not linked (or there are not children) of the previous predicted labels. To this purpose we introduce two masking matrices: Ch_matrix and Sub_matrix

Ch_matrix: is a matrix of dim |Volume_length|*|Chapter_length|. This matrix has rows as many as the number of Unique volumes and Columns as many as the number of Unique Chapters. In other words any row corresponds to a specific volume and any column corresponds to a specific chapter.

Now, let's say that the unique volume is the set of { vol-1, vol-2, …. vol-N } and the unique chapters is the set of { ch-1, ch-2, … ch-N }. Any random selected volume vol-i has children {ch-i1, ch-i2, … ch-ik}. The row of Ch_matrix which correspond to vol-i is everywhere zero, except for the columns which are children of this volume.

So, Ch_matrix[i,j] = 1 **IF** Chapter-j is child of Volume-i **ELSE** 0, or to put it differently,

Example: If Sub_matrix=

	ch-1	...	ch-j	ch-k	ch-h	...	ch-N
vol-1	0		0	0	1		1
vol-2	0		1	0	0		0
...							
vol-N	1		0	1	0		0

Then, this means that chapters {ch-h, ch-N} are children of vol-1, chapter {ch-j} is children of vol-2 and so on.

Now, If for every vol-i we sum it with respect to the Chapter-Columns or *Sub_matrix.Sum(dim=1)* we have the number of children of vol-i, while if we sum each Column-j it with respect to the Rows or *Sub_matrix.Sum(dim=0)* we have in how many volumes is appeared ch-j. This is what the image below shows

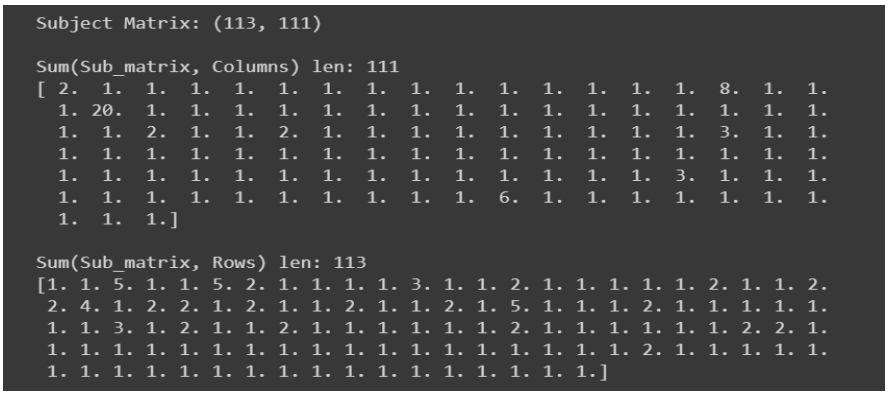
```
# Chapter-Matrix Analysis:
print("Chapter Matrix:", Ch_matrix.shape); print()

print("Sum(Ch_matrix, Columns) len:", len(Ch_matrix.sum(0)),); print(Ch_matrix.sum(0) );

print("Sum(Ch_matrix, Rows) len:", len(Ch_matrix.sum(1)),   ); print(Ch_matrix.sum(1) )
```

```
Chapter Matrix: (39, 113)

Sum(Ch_matrix, Columns) len: 113
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

Sum(Ch_matrix, Rows) len: 39
[2. 5. 8. 1. 6. 5. 3. 3. 5. 2. 2. 3. 3. 6. 5. 1. 5. 4. 2. 2. 2. 2. 1. 2.
 1. 1. 2. 2. 1. 4. 3. 3. 2. 3. 5. 3. 1. 1. 1.]
```

From this image we can infer that, vol-1 has two chapters, vol-2 has five, vol-3 has eight etc. We also infer that all columns belong to one and only one volume.

Sub_matrix: We repeat the same process in order to create a matrix which can mask Subject values based on the Chapters' predictions. So we construct the Sub_matrix which has dimensions |Chapter_length|*|Subject_length| and like the former matrix it is consisted of zeros and ones. In other words:

Sub_matrix[i,j] = 1 **IF** Subject-j is child of Chapter-i **ELSE:** 0



The above image, like previously, has the sum of the matrix if we sum it horizontally and vertically. From this we can infer that ch-1 and ch-2 have one subject, ch-3 has 5 etc. However if we look the sum by column (vertically) we can conclude that there are Subject-Columns which belong to more that one Chapters, for example the Subject ["ΓΕΝΙΚΕΣ ΔΙΑΤΑΞΕΙΣ"] belong to twenty chapters, while the Subject ["ΔΙΑΦΟΡΕΣ ΔΙΑΤΑΞΕΙΣ"] belongs to eight subjects.

In effect this means that Raptarchis dataset is not Tree Graph.

Masking Process:

Having constructed these matrices, we can proceed to the masking process:
First, for any given instance, VOL-NN gives as output a vector of dim [1xVol_length] which corresponds to the predicted volumes for this instance. if we multiply this vector with the Ch_matrix we have a vector v_ch = [1xCh_length]. Every position o this vector correspond to a specific Chapter and hold the Sum-outputs of parent nodes of this Chapter. As we said before, becacuse the a chapter has only one parent the values of this vector will be excatly one of the outputs of
OL-NN. After that the mask v_ch will cover the [1xCh_length] output of CH-NN.
The same proceess will be apply to mask the Subjects. The only difference will be that because of the fact that a subject may have many parent nodes, the ve v_sub = [1xSub_length] vector, which will be resulted from CH-NN and Sub_matrix, will have
 # The matrices
Ch_matrix = [Vol_lengthxCh_length], Sub_matrix = [Ch_length1xSub_length]

 # The outputs of NNs
Vol-pred = [1xVol_length]

Ch-pred = [1xCh_length]
Sub-pred = [1xSubl_length]

 # Masking
ch-mask   = Vol-pred .dot(Ch_matrix) = [1xCh_length]
sub-mask = Ch-pred .dot(Sub_matrix) = [1xSub_length]

 # Outputs:
vol-out = Vol-pred
ch-out = Ch-pred*ch-mask # ElementWise multiplication with ch-mask
sub-out = Sub-pred*sub-mask # ElementWise multiplication with sub-mask

ATTENTION: As we said before it is used in pytorch's CrossEntropyLoss for loss function. That means that there is no need to use an Softmax Activation Function of the output of the NNs. So the masks contain the Raw output of NNs rather than their probability distribution.

SumLast_Bert:
Similar to SimpleBert but instead of [CLS] token we sum all the text tokens of the last layer and afterwards we use them for input to NNs.

SumLast4_Bert:
Similar to SimpleBert but instead of [CLS] token we sum all the text tokens of the last 4 layers and afterwards we use them for input to NNs.

Results

Model	**Loss**	**Acc**	**mF1-Vol**	**mF1-Ch**	**mF1-Sub**
SimpleBert	**1.36** [0.30, 0.48, 0.57]	**0.83** [0.93, 0.89, 0.86]			
Last3_Bert	**1.29** [0.29, 0.45, 0.54]	**0.83** [0.93, 0.89 0.85]			
Mask_simpleBert	**1.98** [0.61, 0.74, 0.63]	**0.77** [0.86, 0.89, 0.85]			
Sum_LastBert	**1.33** [0.30, 0.47, 0.56]	**0.84** [0.93, 0.90, 0.86]	0.91	0.84	0.86
sum_Last4Bert	**1.56** [0.35, 0.55, 0.65]	**0.81** [0.92, 0.87, 0.83]	0.87	0.81	0.85

mF1 stands for Macro-average-f1 score