Data Mining Technology for Business and Society

Homework 3 - Deep Learning and Natural Language Processing

Team Members:

- **Loretucci Lorenzo** (ID. 1903794)
- **Nana Teukam Yves Gaetan** (ID. 1741352)

Preprocessing:

For the processing we decide to take the claims having just one entity with a single token as expressed in the fourth point of the preprocessing steps.

Task 1. Use LAMA to get predictions for masked entities:

1.1) The first thing we did it to modify the function `'get_ranking'` such that we have as result just the first prediction from BERT model. Then we modified the file `'eval_generation.py'` in the following way:

- We added the modified version of the function `'get_ranking'`, then we also added the functions `'__max_probs_values_indices'` and `'__print_top_k'` from the file `'evaluation_matrics.py`.
- We changed the way we parse the sentences to the model, because the initial file took sentence from the terminal. Instead of running the script with a single sentence, we ran the script with the path relative to the file containing the masked sentences. Thus, the model reads the sentences and predict an output, one after the other in an automatic way.
- To optimize the heuristic we decided to convert both the output of BERT model and our true masks in lowercase.
- Once we had the results of the predictions, we then computed the accuracy. To compute the accuracy we used a simple *heuristic*. In case the first prediction matched the mention, we assigned a label SUPPORTS, if not we assigned a label REFUTES. Then we compared these labels with the original labels (coming from the file) to find how many time they did matched.

For this task we used the Dev set. After the preprocessing part, we were left with **1173** relevant claims. Here is our result:

Support	Reject	Accuracy
661	512	56.351%

1.2) Starting from the code created for the task **1.1**, we modified the function `'get_ranking'` such that we have as result the `'top 10 predictions'` from BERT model. Moreover, we also modified the last part of the script. More precisely the part where we compute the accuracy.

For this task the accuracy was computed more or less in the same way as we did in the previous task. The only difference is that in order to assign a SUPPORT or REFUTES, we check if the true masked value is contained in the top 10 predictions.

The input data for this task are the same of previous task. Here is our result:

Support	Reject	Accuracy
684	489	58.312 %

1.3) Also for this task we started from the code created for task **1.1**. We modified the function `'get_ranking'` to get not only the predict word but also the log-probability associated to that word. In order to create Fig.1, we simply computed for each threshold the probability BERT associates with the correct token.

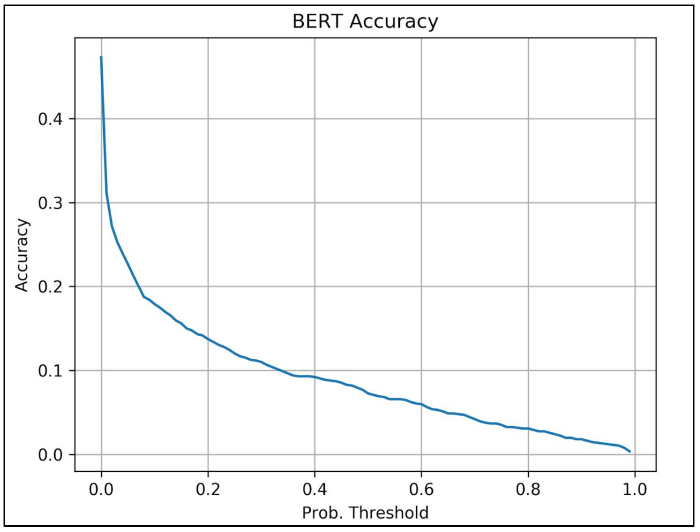

Fig.1. BERT Accuracy.

From this plot we can see that as the predicted probability approaches to 1, the accuracy slowly decreases. From this plot it is not really clear which threshold should be used in order to run a "good" analysis. The "perfect" threshold will depend on the type of analysis you want to do.

Task 2. Train a classifier on MASK and gold representations:

For this task created a new file starting from `'get_contextual_embeddings.py'`.

In order to create the data frames useful for the Training and Prediction, we started by concatenating the contextual representation both masked and unmasked claims thanks to the function "`embed`". Then we concatenated the vectorial results to obtain a DataFrame that we used to run a binary classifier.

For our model we decided to use SVM, but before building it we performed a GridSearch analysis to find the best hyperparameters. We obtained the best result with the following parameters: *{'C':[1000], 'gamma': [0.0001], 'kernel':['rbf'], 'degree': [10]},* testing on the Dev set and having an *accuracy* of **0.6818.**

Then we run the model on the whole Training dataset (number of sentence = **22072**), and used the Test dataset to predict the Labels of our claims. At last we saved the predicted labels to a jsonl file.