Jetbrains AI

Evaluation system – Text formality

Author : Jakub Smolarczyk

1. Documentation

a. Project structure

The Project consists of two python scripts:

- LocalDataset.py – to prepare data from dataset
- main.py – to evaluate the chosen model

b. Dataset

For the model evaluation, the following dataset was used: osyvokon/pavlick-formality-scores. It is made of three columns:

- domain - the domain from which this sentence is taken. This column is not used in evaluation system
- avg_score – score of the sentence, represented by a float value in the range of -3.0 to 3.0, where -3.0 means highly informal sentence and 3.0 means highly formal sentence
- sentence – the sentence to be evaluated

This dataset has approximately 11270 rows, of which 2000 are labeled as test data and 9270 are labeled as training data. Also the values from the avg_score column were converted to match the classification problem instead of the regression one. The negative values were given the „0” class and non-negative values were given the „1” class.

c. Models

In evaluation process the following models were used from s-nlp:

- s-nlp/xlmr_formality_classifier
- s-nlp/roberta-base-formality-ranker
- s-nlp/deberta-large-formality-ranker
- s-nlp/mdeberta-base-formality-ranker
- s-nlp/mdistilbert-base-formality-ranker

All of these models return 1x2 tensor with two values indicating the probabilities of both classes (1 for informal ; 0 for formal) for the given sentence. These classes are inverted before calculating the chosen metrics.

d. Metrics

To assess the performance of the above classifiers, the following metrics were employed:

- Accuracy – measures the proportion of correctly classified instances among the total number of predictions.
- Precision - quantifies the proportion of correctly predicted positive instances out of all predicted positives.
- Recall - measures the proportion of actual positive cases that were correctly identified by the model.
- F1 Score – functions as a harmonic mean of precision and recall, providing a balance between the two metrics.
- Log loss - measures the uncertainty of predicted probabilities relative to the actual class labels.
- ROC-AUC (Receiver Operating Characteristic - Area Under Curve) - evaluates how well the classifier distinguishes between classes by plotting the true positive rate against the false positive rate at various threshold settings.

All of these metrics exceept Log loss have a range of 0.0 to 1.0, where the higher the value, the better the performance of the model. Log loss is not as limited as other metrics and can have any positive value, where 0.0 indicates perfect performance of the model.

e. Tools

In this Project the PyCharm IDE was used to write python scripts and the following libraries were installed:

- PyTorch – to simplify the processing of tensors in a loaded model
- SciKit-learn – used to calculate the evaluation metrics
- Transformers – used to load tokenizers and models
- Csv – to save the results in a .csv file

2. Report

The most challenging part for me was the research process. Most of the datasets that I have found were not publicly available, so I kept searching until I have found the Hugging Face, from where I have searched all datasets and text classification models. To simplify the scripts, I chose models that give the same output (Tensor with a shape of 1x2). Also it took me some time to understand PyTorch and transformers functionalities. At first I tried to use spaCy library to tokenize the input – sentences from the dataset, but then I realized that the same process can be done easier in the context of my project using transformers. Using scikit functions for the metrics calculation was an easy task, because the names of those functions and their parameter lists make their purpose clear.

s-nlp models	Evaluation metrics					
	accuracy	precision	recall	F1 score	Log loss	ROC AUC score
Xlmr formality classifier	0,712	0,667	0,971	0,791	1,325	0,841
	0,710	0,663	0,961	0,785	1,323	0,841
Roberta base formality ranker	0,460	0,518	0,568	0,541	1,688	0,556
	0,454	0,505	0,561	0,531	1,704	0,556
Deberta large formality ranker	0,553	0,587	0,693	0,635	1,435	0,665
	0,550	0,577	0,688	0,628	1,449	0,667
Mdeberta base formality ranker	0,524	0,586	0,520	0,551	1,816	0,600
	0,524	0,577	0,517	0,545	1,808	0,602
Mdistilbert base formality ranker	0,562	0,611	0,607	0,609	1,708	0,645
	0,562	0,603	0,604	0,603	1,703	0,648

Table 1. Results obtained during evaluation of all the models from this documentation. The results of model evaluation with only test rows (2000) from dataset are marked in green, while evaluation with both test and train rows (2000 + 9270) from dataset are marked in cyan.

In the above table 1, I have included the results obtained during the evaluation of all the models listed in the documentation. Firstly, the dataset test and train groups are well balanced between each other, because of the similar results in both cases. Next we can see that the first model performed the best. All of the metrics except the log loss are the highest for this model. This combined with the lowest value of log loss indicates that this model is the best choice from s-nlp models in terms of detecting formality in texts. On the other hand the second model performed the worst. It is the only model that achieved an accuracy of less than 50% and has an ROC AUC score of less than 0,6, which means that this model's prediction were mostly random (0,5 would imply total randomness). This might also explain why the log loss value for this model is not the worst, compared to other models. Because of the randomness, most of the predictions were uncertain (both positive and negative), thus lowering the final log loss value.