README | MAYN Semantic Relations Software Project

This Github repository contains all code and data Team MAYN used for their Semantic Relations Software Project. The README has the same structure as the repository and serves as a short guide through the project, explaining the purpose of all code and data files in the repository.

Codebase

Codebase contains all code used in our project. We worked mainly in Python, sometimes through Jupyter Notebooks. Performance Analysis (mainly advanced confusion matrices) was done in R. We utilized compute through Google Colab for our Fine-Tuning, but replications can be done with all fitting compute providers.

Preprocessing

In preprocessing there are two scripts which we have used to restructure our raw data so that we can use it further. This alsoour guideline to verb masking and our masked sentences

Satz_umformer.py: Script to reformulate our sentences and extract verbs
Semeval_reader.py: Script to parse the TRAIN_FILE-sentences in a list
Masking Guidelines: Annotator Guidelines for which verb to mask in the manually masked sentences,
Test_sentences.txt: The masked sentences

Zero-Shot:

Zero-shot contains all the scripts that we used for our zero shot testing.

Performance

Contains our classification algorithm, two scripts for final testing and the notebook to get the zero-shot results

algo.md: The algorithm used for the zero-shot testing
Bert_predict_class.py: Script to feed the masked sentences into BERT and to predict the class
Softmax_and_log_dicts.py: contains the dictionary of each class with the frequency of their verbs and also the softmax and log function
LOG_and_SOFTMAX_result.ipynb: evaluating our masked sentences and getting our zero-shot results

Interpretability

The notebook we used to obtain and visualize interpretability results for BERT

Token_importance_visualizer.ipynb: Script which replaces each word from the sentence with an UNK token and finally returns the sentence with each word colored in a certain shade. Deep red coloured words are the words which contributed **most** to the determination of the class and white/light coloured words are **less** important.

Fine-Tuning:

Fine-Tuning contains all the scripts we used for our fine-tuning and FT testing.

Performance

Contains our notebook for evaluating the sentences with a NN-model

Semeval_multiclassification.ipynb: Our NN-model for this project

Interpretability

Our notebook to get interpretability results for our fine-tuning model

Yi-Wan Code

Data

Data contains all the results we got from our code

SemEval

In the SemEval folder, all the data we pulled from the original SemEval-2010 Task 8 here are included. We use training data to extract verbs for evaluating our zero-shot MLM and to fine-tune our Multi-Classifier Neural Network. We use testing data to evaluate performance.

Zero-Shot

Zero-shot contains all the results from zero-shot testing

Performance

Contains zero-shot results using two variations of our algorithm (logarithmized and softmaxed) and confusion matrices that analyse respective performance.

Zero Shot Results Log csv: zero-shot results with log
Zero Shot Results Softmax csv: zero-shot results softmaxed
Confusion Matrix LOG ZS rmd: confusion matrix for the log results

Confusion Matrix Softmax ZS: confusion matrix for the softmax results

Interpretability

Interpretability
Results which were used for qualitative interpretability in the report

Interpretability zero-shot.pdf: cherrypicked sentences to analyse the interpretability from our zero-shot model

Fine-Tuning:

Fine-Tuning contains all the results from Fine-Tuned testing

Performance

Contains fine-tuning results and confusion matrix

Neural_net_results.csv: Our NN-model results
Confusion Matrix NN Fine-Tuned.rmd: confusion matrix for the NN performance

Interpretability

Results used for interpretation

Yi-Wan shapley data: cherrypicked sentences to analyse the interpretability from our NN model