

CSI5386: Natural Language Processing

Assignment 2: Text classification using deep learning

Group Members

- **300311048: Abhisht Makarand Joshi**
- **300279256: Nidhi Kumari Chauhan**
- **300295129: Mohana Prriya Sivakumar**

Assignment 2 tasks were equally divided and assigned to each team member.

1. Team member 1

- a. Concatenating all text files into a corpus
- b. Legal text classification: Performing three pretrained models.

2. Team member 2

- a. Concatenating all text files (merging datafiles for training and testing dataset)
- b. PreProcessing Dataset for the model evaluation
- c. Hypertuning Optimzation on different HyperParameters
- d. Evaluation of model: Sbert
- e. Legal text classification: three own models.

3. Team member 3

- a. Hypertuning Optimzation on different HyperParameters for models
 - b. Evaluation of model: SimCSE
 - c. Calculating Pearson Correlation for all model
- all-mpnet-base-v2
 - multi-qa-mpnet-base-dot-v1
 - sup-simcse-roberta-large
 - sup-simcse-bert-large-uncased

After collaborating all the work done by the respective team members, each one of us have understood and evaluated part 1 and part 2 of the assignment together along with the analysis of the report.

PART 1 (Legal text classification)

Legal text classification is deployed to identify the category of a legal text based on the association between the legal text and that category using deep learning technologies. The dataset containing 510 text files is extracted and combined for preprocessing upon which further processing is handled. The goal is to extract and categorize the corpus material into the 41 classes, for which the code along with trained model is provided[1].

Among all the available pretrained models[4] to implement the same and obtain the Area under the PR curve (AUPRC) metric, three of the following models were deployed[6].

1. deberta-v2-xlarge (3.1GB)
2. roberta-base (447.4MB)
3. roberta-large (1.3GB)

To acquire performance metrics like Precision, Recall, F-score, and Area under the PR curve(AUPR), the stated pretrained models that are already available and fine-tuned are run. However, the AUPRC code is mainly considered as the deciding factor.

Apart from the above models, three more models of our own, listed as following is deployed and fine tuned, respectively.

1. bert-base-uncased
2. distilroberta-base
3. albert-base-v2

Before the aforementioned models were trained and tested, a significant problem emerged. Due to the size of the provided data, the "CUDA out of memory" issue appeared regardless of whether the model was deployed on a local machine or through a cloud-based service like kaggle and Google colab.

The data is trimmed for both the training and testing data at the appropriate ratios of 80:20 (88 training files and 22 testing files) to handle the out-of-memory error. Even after the post-data trimming process, we still encountered sentences that were extremely long. To address this issue, sentences are chosen for each data set based on the context and the number of words in each sentence.

Among the all six models listed, the pretrained models were deployable in the local computer, however, the other three own models were trained and tested on kaggle only using High power GPUs.

The AUPR measure is then calculated by applying the modified data set to each of the six models mentioned above.

	deberta-v2-xlarge	roberta-base	roberta-large	bert-base-uncased	distilroberta-base	albert-base-v2
AUPR value	0.478	0.426	0.482	0.003	0.401	0.103

Table: AUPR Results

Albert-base-v2

```
gt_dict = load_json("/kaggle/working/cuad-main/data/test_final.json")
gt_dict = get_answers(gt_dict)
len(gt_dict.keys())

results = get_results("./train_models/albert-base-v2/", gt_dict, verbose=True)
```

```
123
AUPR: 0.103, Precision at 80% Recall: 0.000, Precision at 90% Recall: 0.000
```

distilroberta-base

```
gt_dict = load_json("/kaggle/working/cuad-main/data/test_final.json")
gt_dict = get_answers(gt_dict)
len(gt_dict.keys())

results = get_results("./train_models/distilroberta-base/", gt_dict, verbose=True)
```

```
861
AUPR: 0.401, Precision at 80% Recall: 0.256, Precision at 90% Recall: 0.077
```

bert-base-uncased

```
gt_dict = load_json("/kaggle/working/cuad-main/data/test_final.json")
gt_dict = get_answers(gt_dict)
len(gt_dict.keys())

results = get_results("./train_models/bert-base-uncased/", gt_dict, verbose=True)
```

```
123
AUPR: 0.003, Precision at 80% Recall: 0.000, Precision at 90% Recall: 0.000
```

roberta-large

```
gt_dict = load_json("./data/test.json")
gt_dict = get_answers(gt_dict)
len(gt_dict.keys())
results = get_results("./pre_trained/roberta-large/", gt_dict, verbose=True)
```

```
AUPR: 0.482, Precision at 80% Recall: 0.381, Precision at 90% Recall: 0.000
```

roberta-base

```
gt_dict = load_json("./data/test.json")
gt_dict = get_answers(gt_dict)
len(gt_dict.keys())
results = get_results("./trained_models/", gt_dict, verbose=True)
```

AUPR: 0.426, Precision at 80% Recall: 0.311, Precision at 90% Recall: 0.000

deberta-v2-xlarge

```
gt_dict = load_json("./data/test.json")
gt_dict = get_answers(gt_dict)
len(gt_dict.keys())
results = get_results("./deberta-v2-xlarge/deberta-v2-xlarge/", gt_dict, verbose=True)
```

AUPR: 0.478, Precision at 80% Recall: 0.440, Precision at 90% Recall: 0.178

As a result, the **pretrained model roberta-large** on the entire dataset had the highest AUPR score, which was **0.482**. Additionally, the **own model, distilroberta-base** received the highest **AUPR score of 0.401** for the own category of models that was trained, tested and fine tuned in the ratio of 80:20, or 88 training files and 22 testing files.

PART 2 (Training sentence similarity models)

Sentence embedding uses vectors to represent complete sentences and the semantic data they include. This makes it easier for the machine to understand the context, goal, and other intricacies of the complete text.

In assignment 1, the **SBert model** - 'all-MiniLM-L6-v2' had achieved the best Pearson Correlation value for three input files (Headline, Postediting, Questions) and for the files (Answer and Plagiarism), the **SimCSE model** - 'sup-simcse-bert-base-uncased' had achieved the best Pearson correlation value.

Hence, as a team it was decided to fine tune and evaluate the parameters using `word_embedding_model.get_word_embedding_dimension()` function for both models.

A brief about Dataset:

To train for a few Sbert and SimSCE models, we first merged the Semantic Textual Similarity (STS) data from 2012, 2013, 2014, and 2015, but the computation took more than 12 hours. In order to examine and analyse more models, we therefore reduced the training dataset and

used data from 2012.

Semeval 2016-Task1 Semantic Textual Similarity (STS) dataset, which was also utilised for assignment 1, was used for testing. But because we were testing new models, we combined all the files (headlines, postediting, answers, questions, and plagiarism) and then performed a new Pearson's correlation analysis on the combined file.

The data was initially preprocessed just like we did in Assignment 1 and then used for both training and testing purposes.

The results obtained with the mentioned models is as following,

1. *SimCSE Model - "sup-simcse-bert-base-uncased"*

We experimented with combinations of epochs [1 to 10] and batch_size [16,32,64] however, the pearson correlation value remained constant and fell in the range between 0.01 and 0.2%.

2. *Sbert Model - "all-MiniLM-L6-v2"*

For this model, after hyperparameter tuning, an improvement in the Pearson correlation value was observed increasing by 0.012%, using the parameters: epochs=4, batch_size=64, shuffle=True

	Before tuning	After tuning
Pearson correlation value	0.75668	0.75680

Table: Sbert Pearson Correlation Before and After Fine Tuning

However, we chose to modify the models and reevaluate them in light of the fact that the improvement was minimal. Hence, proceeding further, the following models were considered.

1. all-mpnet-base-v2
2. multi-qa-mpnet-base-dot-v1
3. sup-simcse-roberta-large
4. sup-simcse-bert-large-uncased

The pearson correlation was dropped in almost all of the forementioned models, with the exception of the following models with the given parameters, on which hypertuning was manually performed using a combination of **epochs [1 to 10]** and **batch size [16, 32, 64]**.

Model	Parameters	PC value before tuning	PC value After tuning	Improvement percentage
all-mpnet-base-v2	Epoch: 4, Batch size: 64	0.77775	0.78049	0.274%
all-mpnet-base-v2	Epoch: 3, Batch size: 64	0.77775	0.77875	0.001%
all-MiniLM-L6-v2	Epoch: 4, Batch size: 64	0.75668	0.75680	0.012%

Table: Pearson Correlation Before and After Fine Tuning for all models

Screenshot for **all-mpnet-base-v2**, parameter: **Epoch: 4, Batch size: 64**

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

D:\Ottawa\CSI 5386 NLP\Nidhi>perl correlation-noconfidence.pl gsmerge.txt Sbert_all-mpnet-base-v2.txt
Pearson: 0.77775

D:\Ottawa\CSI 5386 NLP\Nidhi>perl correlation-noconfidence.pl gsmerge.txt Fine_Sbert_all-mpnet-base-v2.txt
Pearson: 0.78049

D:\Ottawa\CSI 5386 NLP\Nidhi>

```

Screenshot for **all-mpnet-base-v2**, parameter: **Epoch: 3, Batch size: 64**

```

Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

D:\Ottawa\CSI 5386 NLP\Nidhi>perl correlation-noconfidence.pl gsmerge.txt Sbert_all-mpnet-base-v2.txt
Pearson: 0.77775

D:\Ottawa\CSI 5386 NLP\Nidhi>perl correlation-noconfidence.pl gsmerge.txt Fine_sbert_all-mpnet-base-v2_ep3bz64
Pearson: 0.77875

```

Screenshot for **all-MiniLM-L6-v2**, parameter: **Epoch: 4, Batch size: 64**

```

D:\Ottawa\CSI 5386 NLP\Nidhi>perl correlation-noconfidence.pl gsmerge.txt sbert_all-MiniLM-L6-v2
Pearson: 0.75668

D:\Ottawa\CSI 5386 NLP\Nidhi>perl correlation-noconfidence.pl gsmerge.txt Fine_all-MiniLM-L6-v2
Pearson: 0.75680

D:\Ottawa\CSI 5386 NLP\Nidhi>

```

Hypertune Optimization Code Snippet on all-mpnet-base-v2 model, parameter: Epoch: 4, Batch size: 64:


```
# Applying model using CLS pooling and HyperTunning Parameters

model_name = 'sentence-transformers/all-mpnet-base-v2'
word_embedding_model = models.Transformer(model_name, max_seq_length=32)
pooling_model = models.Pooling(word_embedding_model.get_word_embedding_dimension())
model = SentenceTransformer(modules=[word_embedding_model, pooling_model])

train_dataloader = DataLoader(train_data, batch_size=64, shuffle=True)
train_loss = losses.MultipleNegativesSymmetricRankingLoss(model)

model.fit([
    train_objectives=[(train_dataloader, train_loss)],
    epochs=4,
    show_progress_bar=True,
])

#Saving the model
model.save('output/stsb-mpnet-base-v2')
```

Downloading: 100%  571/571 [00:00<00:00, 15.4kB/s]

Result Analysis:

After evaluating multiple models with different combinations of the parameters, we have obtained the set of models which contribute significant improvement in the Pearson correlation value. Among the three models listed above, the “all-mpnet-basev2” with the parameter’s epoch: 4 and batch size: 64 performs the best and showed an improvement by 0.274% respectively.

Result:

Datasets	Pearson correlation before fine-tuning	Pearson correlation after fine-tuning
STS Training data	0.77775	0.78049

With the Pearson correlation value of 0.78047, which is an **improvement of 0.274%** over the initial result, the **Sbert model** performs the best overall after fine-tuning, according to the results of the obtained Pearson correlation values.

Additional References

1. <https://github.com/TheAtticusProject/cuad>
2. https://www.sbert.net/docs/pretrained_models.html
3. <https://github.com/princeton-nlp/SimCSE#model-list>
4. <https://huggingface.co/models>
5. <https://arxiv.org/abs/2103.06268>
6. <https://zenodo.org/record/4599830#.Y4gEWnbMK3A>
7. <https://aclanthology.org/D19-1410/>
8. <https://www.sbert.net/>
9. <https://alt.qcri.org/semeval2016/task1/>
10. https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder
11. https://www.sbert.net/examples/unsupervised_learning/SimCSE/README.html