LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

fine-tuning Bert&S-Bert for clustering and classification

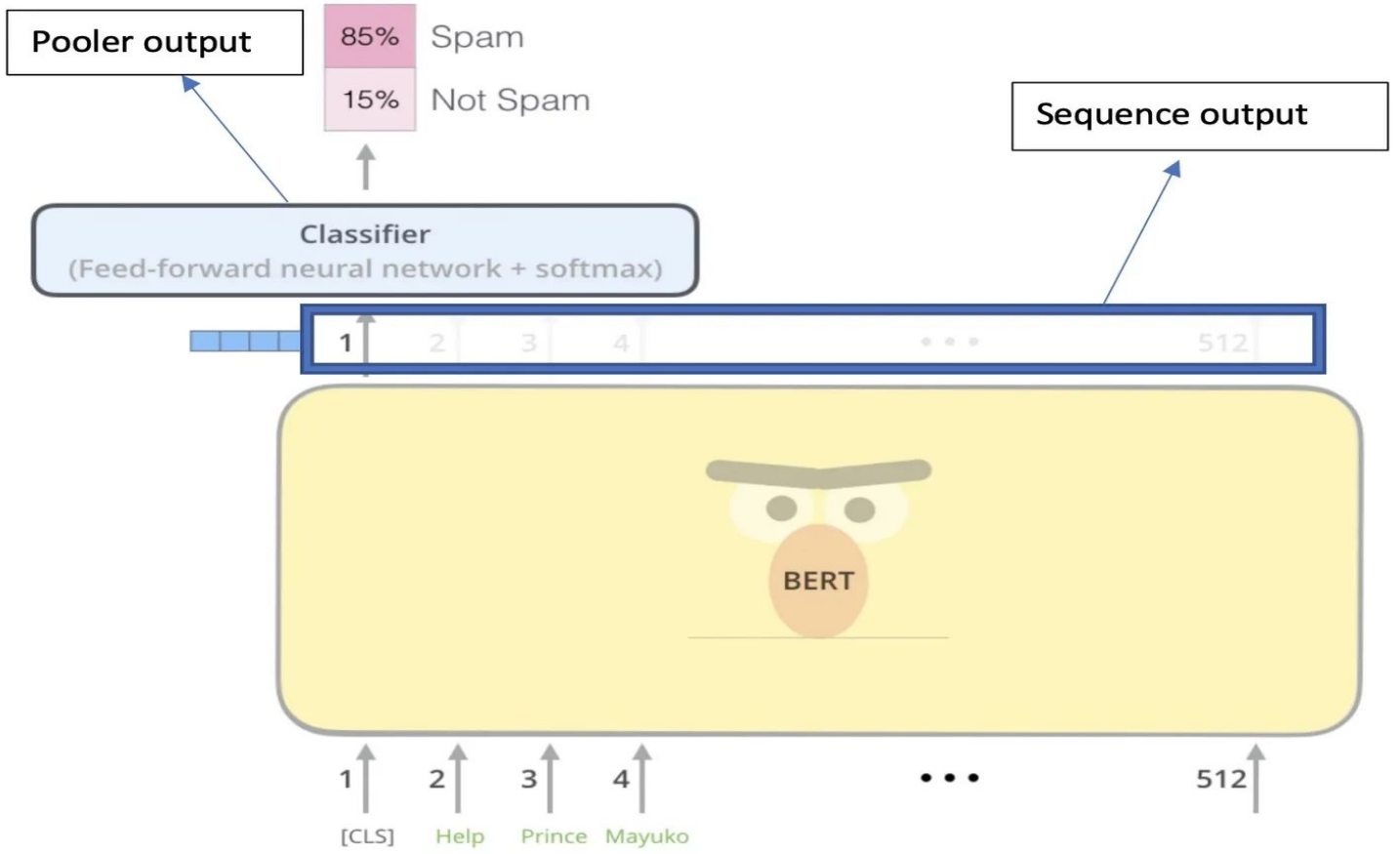
BertForSequenceClassification

Wei Xia, Kaiwei Lei

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Architecture: two outputs in Bert

- The first token of every sequence is a special classification token ([CLS])
- Sentences are separated using a special token ([SEP])
- Token, segment, and position embeddings are summed to produce the final input representation



LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Pre-training BERT

- Two unsupervised tasks:
  - Masked LM and Next Sentence Prediction.

- Masked LM:
  - Randomly masks tokens and the model predicts the masked word.

- Next Sentence Prediction:
  - Teaches the model to understand the relationship between sentences.

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Fine-tuning BERT

- **Process:**
  - Adjusts model parameters for specific tasks using labeled data.
  - Input/output adjustments based on the task (e.g., classification,question answering).

- Fine-tuning is performed on a variety of downstream tasks like text classification, question answering.
- Only requires one additional output layer for the task.

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Process-Preprocessing Data for BERT

- **input ids and attention masks**
  - input_ids: Numeric representation of text.
  - attention_masks: Differentiating real tokens from padding.

```
def process_dataset(dataset, tokenizer, mapping, max_seq_length=128):
    input_ids, attention_masks, labels = [], [], []
    for sample in dataset:
        encoded_dict = tokenizer.encode_plus(
            sample["text"],
            add_special_tokens=True,
            max_length=max_seq_length,
            padding='max_length',
            return_attention_mask=True,
            return_tensors='pt',
            truncation=True
        )
        input_ids.append(encoded_dict['input_ids'])
        attention_masks.append(encoded_dict['attention_mask'])
        labels.append(mapping[sample[key]])
    return torch.cat(input_ids, dim=0), torch.cat(attention_masks, dim=0),
torch.tensor(labels)
```

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Process-Loading and Configuring BERT

- Loading BERT for classification (BertForSequenceClassification)
- Setting up the device for training (CPU/GPU)
- Batch processing of data using DataLoader

```
# Initialize tokenizer and model
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased',
num_labels=num_classes)

train_data = process_dataset(train_dataset)
train_dataloader = DataLoader(train_data, ...)
```

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Process-Training the BERT Model

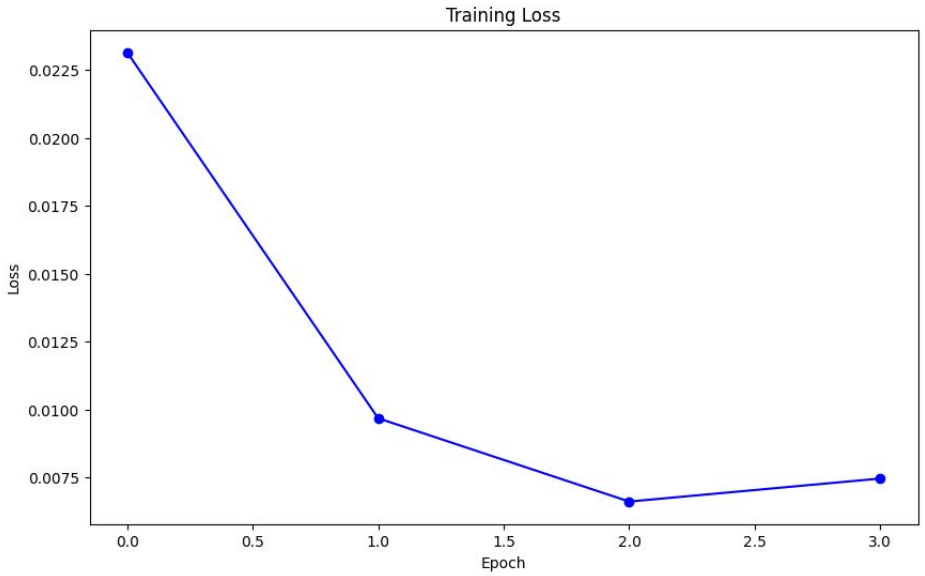
- Iterating over training data in each epoch.
- Model receives `input_ids` and `attention_masks`, outputs probabilities for each category.
- Loss calculation by comparing predictions with actual labels and updating model weights through backpropagation.

```
# Train model
for epoch in range(epochs):
    for batch in train_dataloader:
        outputs = model(input_ids, attention_mask, labels)
        loss = outputs.loss
        loss.backward()  # Backpropagation
        optimizer.step()  # Update weight
```

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Evaluation(letter-language)

Accuracy: 0.9985658676500717



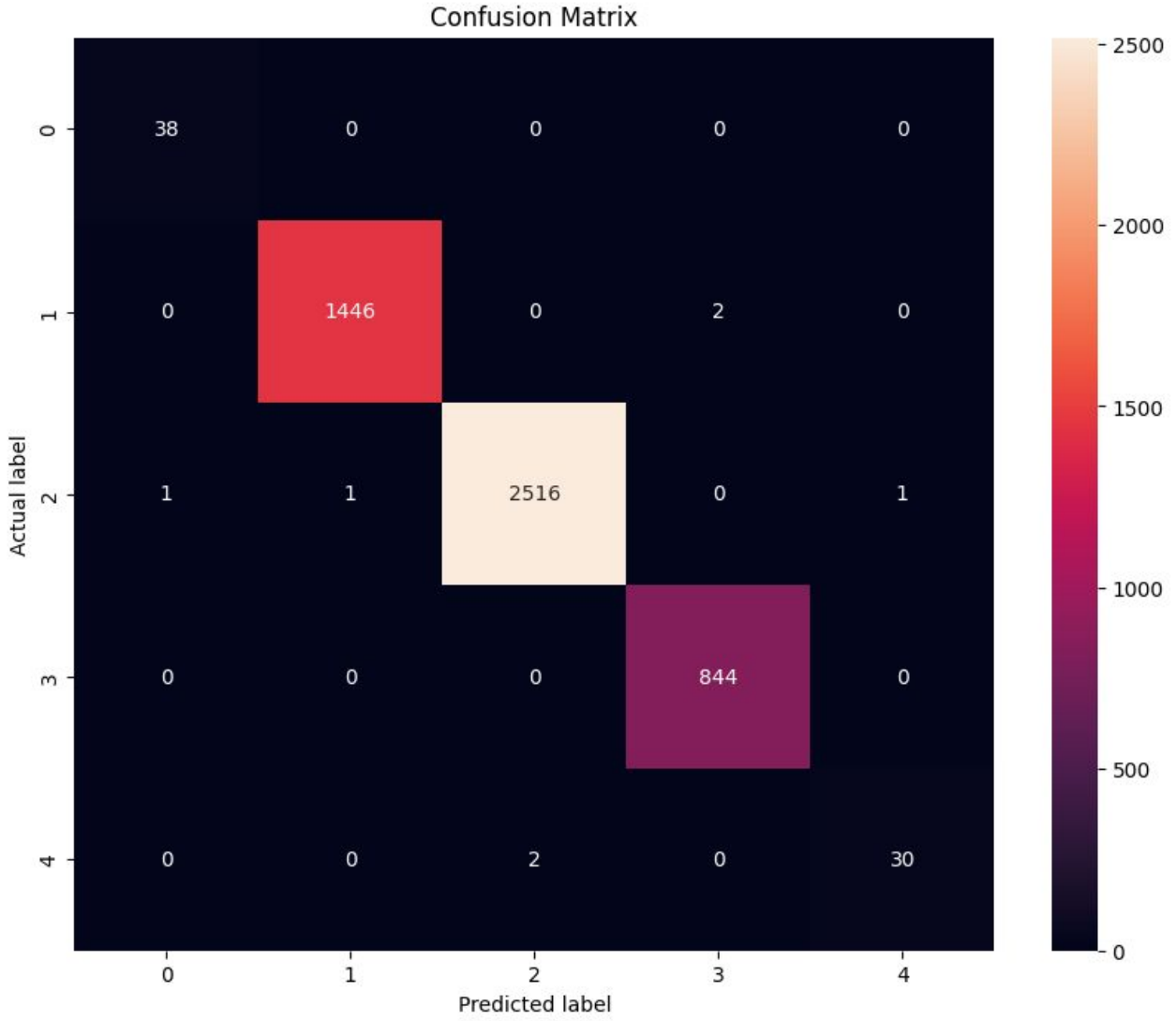
fr

de

en

da

it



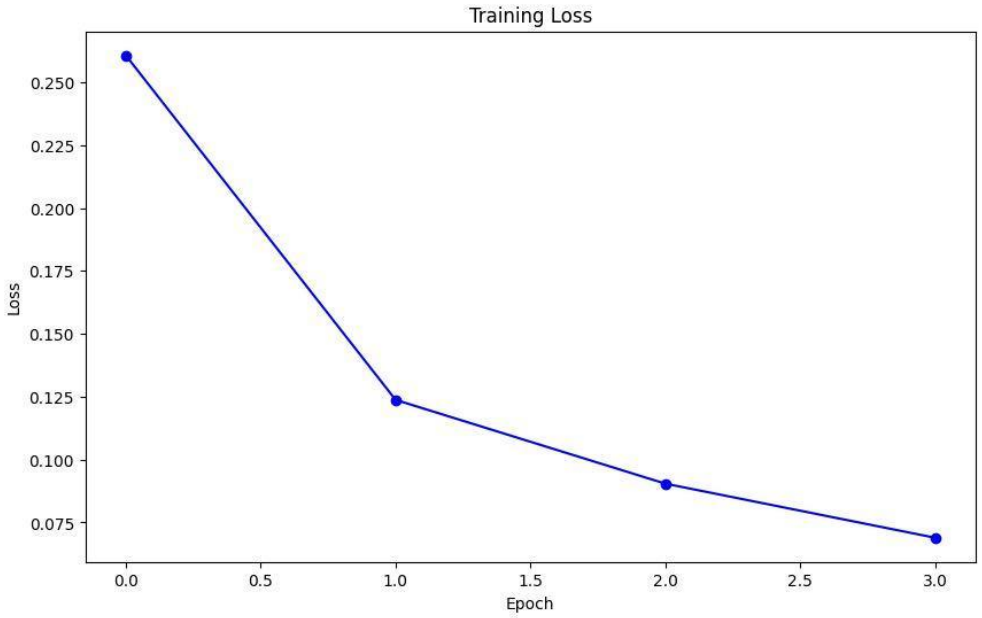
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

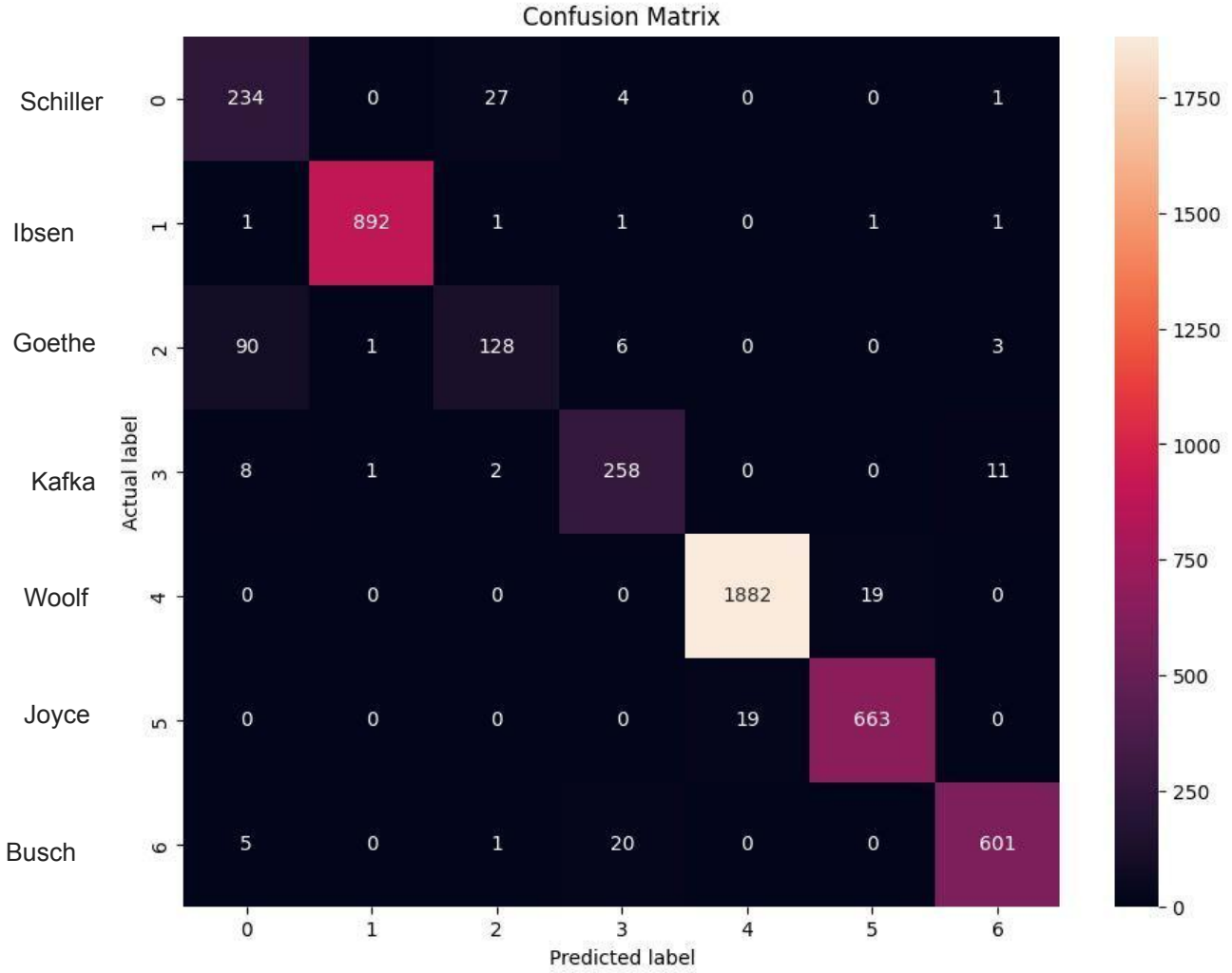
Evaluation(letter-author)

	precision	recall	f1-score	support
Schiller	0.69	0.88	0.77	266
Henrik Ibsen	1.00	0.99	1.00	897
Goethe	0.81	0.56	0.66	228
Kafka	0.89	0.92	0.91	280
Woolf	0.99	0.99	0.99	1901
Joyce	0.97	0.97	0.97	682
Wilhelm Busch	0.97	0.96	0.97	627
Accuracy			**0.95**	**4881**

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Evaluation(letter-author)

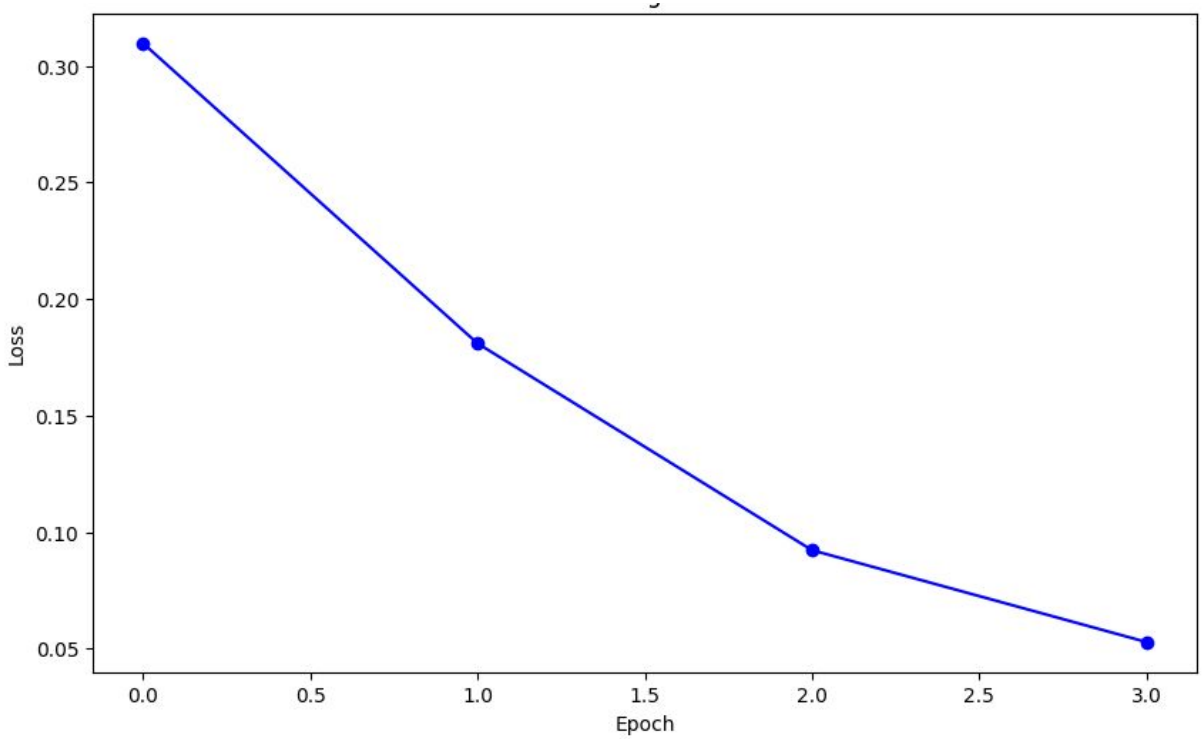


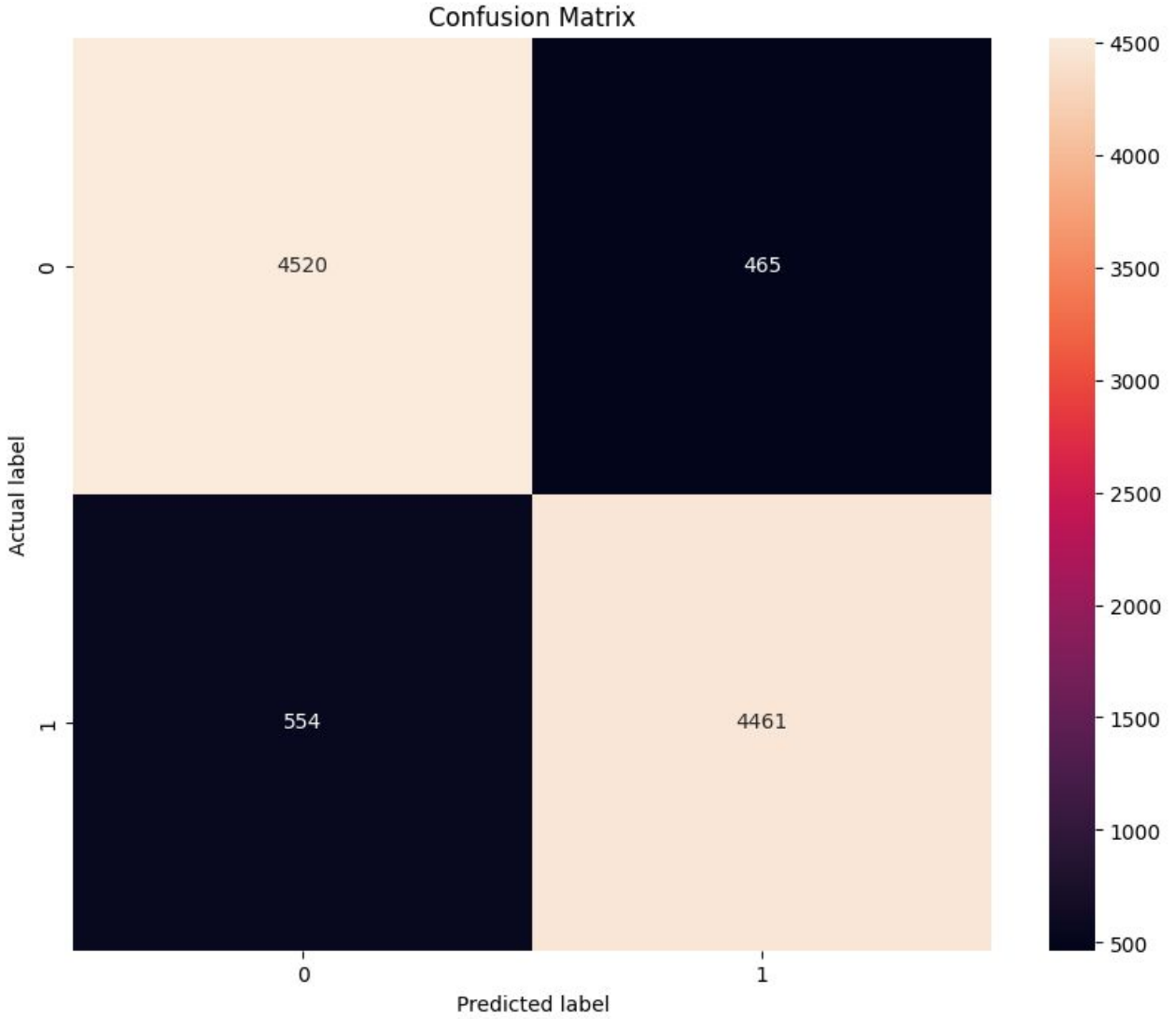


LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Evaluation(sentiment)

	precision	recall	f1-score	support
negative	0.89	0.91	0.90	4985
positive	0.91	0.89	0.90	5015
Accuracy			0.90	10000





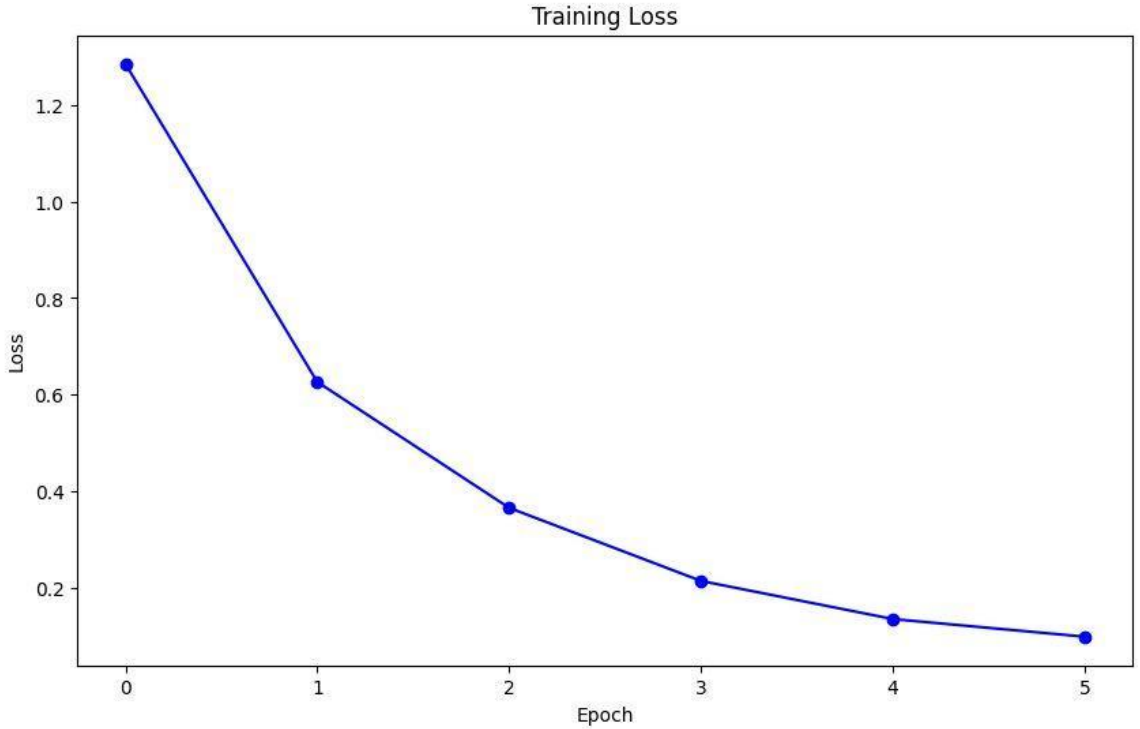
Evaluation(news)

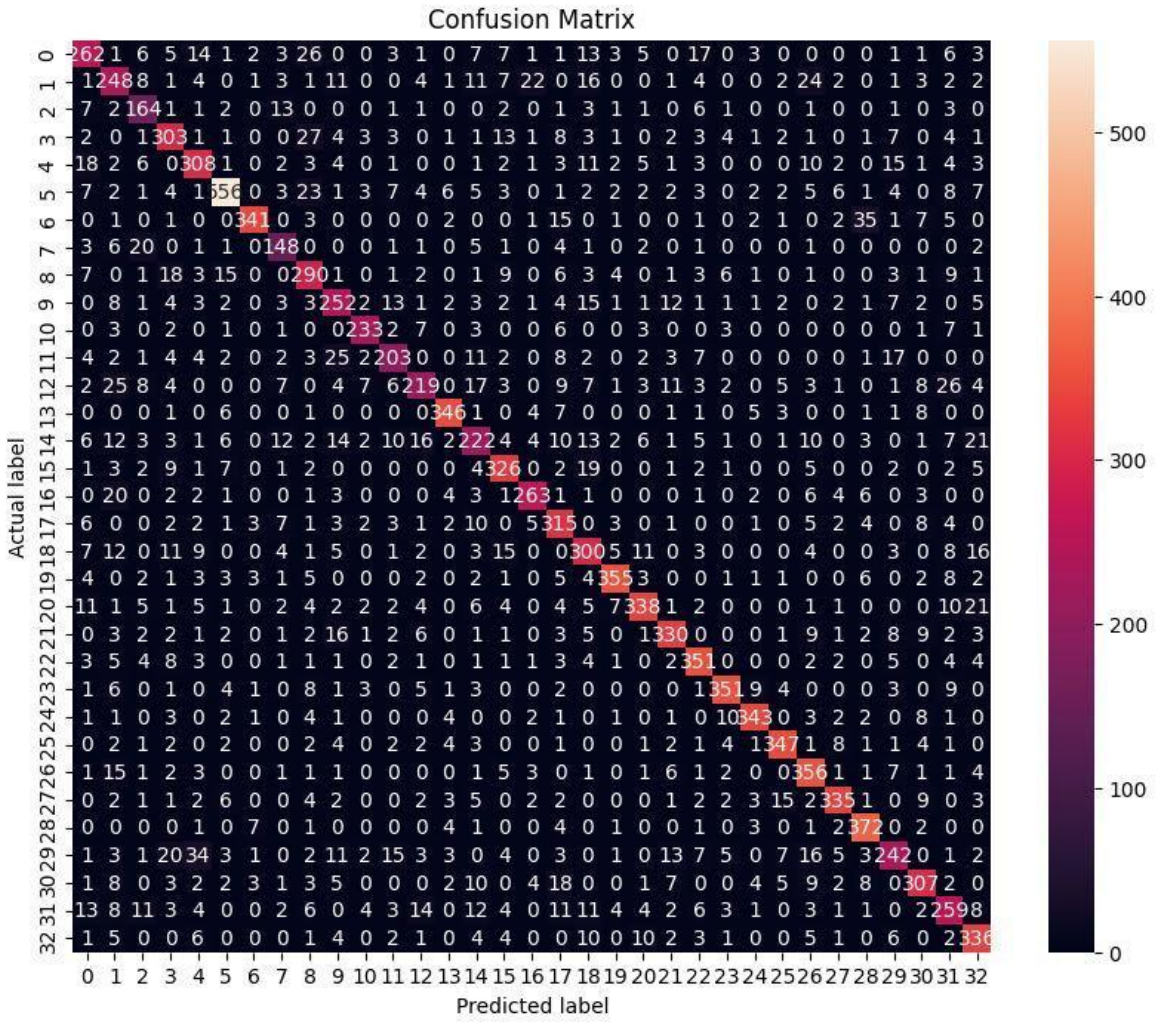
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

	precision	recall	f1-score	support
BLACK VOICES	0.71	0.67	0.69	392
BUSINESS	0.61	0.65	0.63	380
COLLEGE	0.66	0.77	0.71	212
COMEDY	0.72	0.76	0.74	399
CRIME	0.74	0.75	0.74	409
CULTURE & ARTS	0.89	0.83	0.85	673
DIVORCE	0.94	0.81	0.87	419
EDUCATION	0.68	0.75	0.71	198
ENTERTAINMENT	0.67	0.75	0.71	387
ENVIRONMENT	0.67	0.71	0.69	355
FIFTY	0.88	0.85	0.86	273
GOOD NEWS	0.72	0.67	0.69	305
HEALTHY LIVING	0.73	0.57	0.64	386
HOME & LIVING	0.89	0.90	0.90	386
IMPACT	0.62	0.56	0.59	400
MEDIA	0.77	0.83	0.80	395
MONEY	0.83	0.81	0.82	324
PARENTING	0.69	0.81	0.74	391
POLITICS	0.67	0.71	0.69	420
QUEER VOICES	0.90	0.86	0.88	415
RELIGION	0.84	0.77	0.80	440
SCIENCE	0.82	0.80	0.81	414
SPORTS	0.80	0.86	0.83	410
STYLE	0.88	0.85	0.87	413
STYLE & BEAUTY	0.90	0.88	0.89	391
TASTE	0.87	0.87	0.87	397
TECH	0.74	0.86	0.79	416
TRAVEL	0.88	0.83	0.85	405
WEDDINGS	0.83	0.93	0.88	400
WEIRD NEWS	0.72	0.59	0.65	408
WELLNESS	0.79	0.75	0.77	407
WOMEN	0.66	0.65	0.65	400
WORLD	0.74	0.83	0.78	404
Accuracy	**0.77**			**12824**

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Evaluation(news)





LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Sentence Bert (SBERT)

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Agenda

- Overview
- Architecture
- Evaluation
- Discussion

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

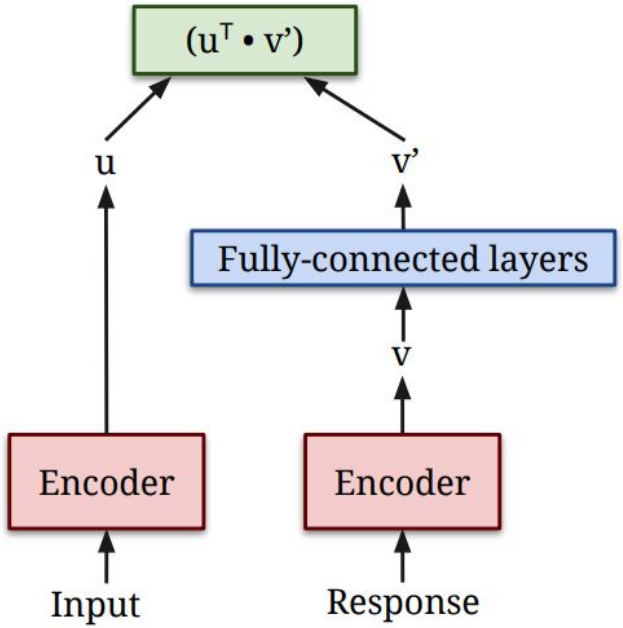
Overview

- Sentence-Bert (SBERT) is a modification of the BERT network
  - This enables BERT to be used for certain new tasks that are up to now not available
  - Including cos-similarity comparison, clustering and IR semantic search
  - As we have seen, through bert encoder, every sentence is mapped to a vector space such that similar sentences are close
  - But bert does not produce sentence embeddings (esp. in cross-encoding contexts)
  - Most common approach to derive bert based fixed-size sentence embeddings is to add an average pooling layer, or just take the first “[CLS]” token
    - (The use of “[CLS]” and “[SEP]”: why does it work?)
- sbert ist trained on NLI data

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

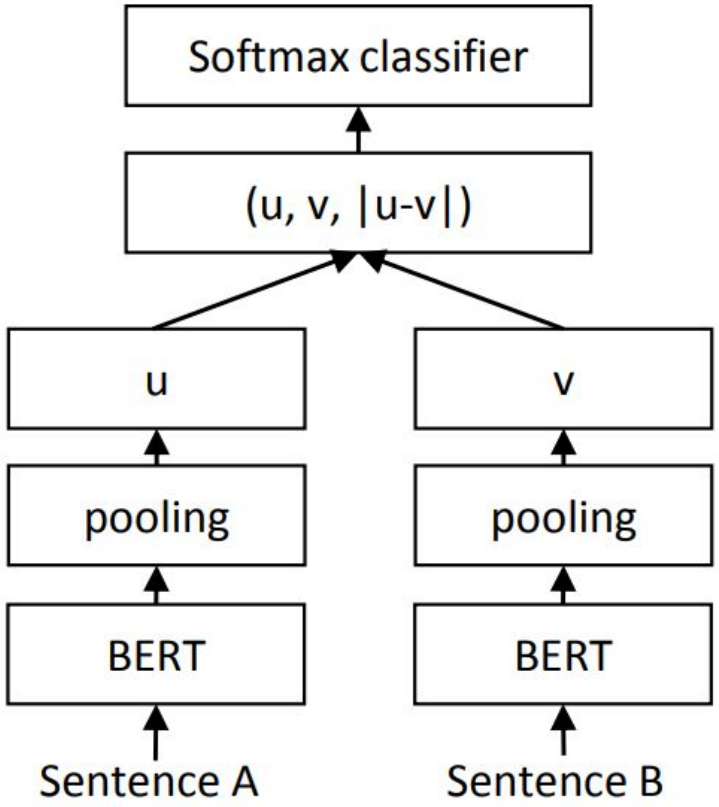
Overview

- sbert utilizes a siamese network:
  - In a nutshell, it consists of two identical artificial neural networks each capable of learning the hidden representation of an input vector, they work in tandem and compare their outputs at the end, usually through a cosine distance. (Chicco, D. 2021)
  - Another example of a siamese architecture from Yang, Y. et al. (2018)
  - Trained on reddit posts consisting of Q and A pairs
- It is argued that due to the parallel nature of training (vis. bert), it achieves significant gains on computational efficiency
- Example: fine-tuning bert for clustering for 10,000 sent:
  - There are 10000C2 combinations (ca. 50m)
  - for sbert it is one embedding for one sentence
- Different architectures tailored for different objectives



LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Architecture

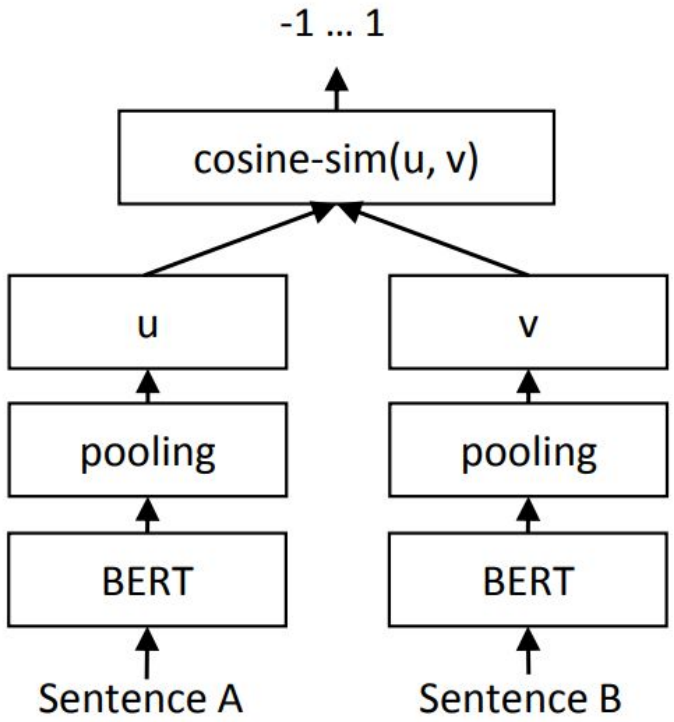


$$o = \text{softmax}(W_t(u, v, |u - v|))$$

- SBERT with classification objective:
  - A softmax classifier is added to the end of the siamese architecture
  - Sentence embeddings u and v are concatenated with the difference |u-v| (trainable weights having the shape 3n x k, where k being the number of labels and n the dimensions of the embeddings)
  - Cross-entropy loss is optimised

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Architecture



- SBERT with regression objective:
  - Cosine-similarity is computed between the two embeddings u and v
  - Loss function is simply MSE

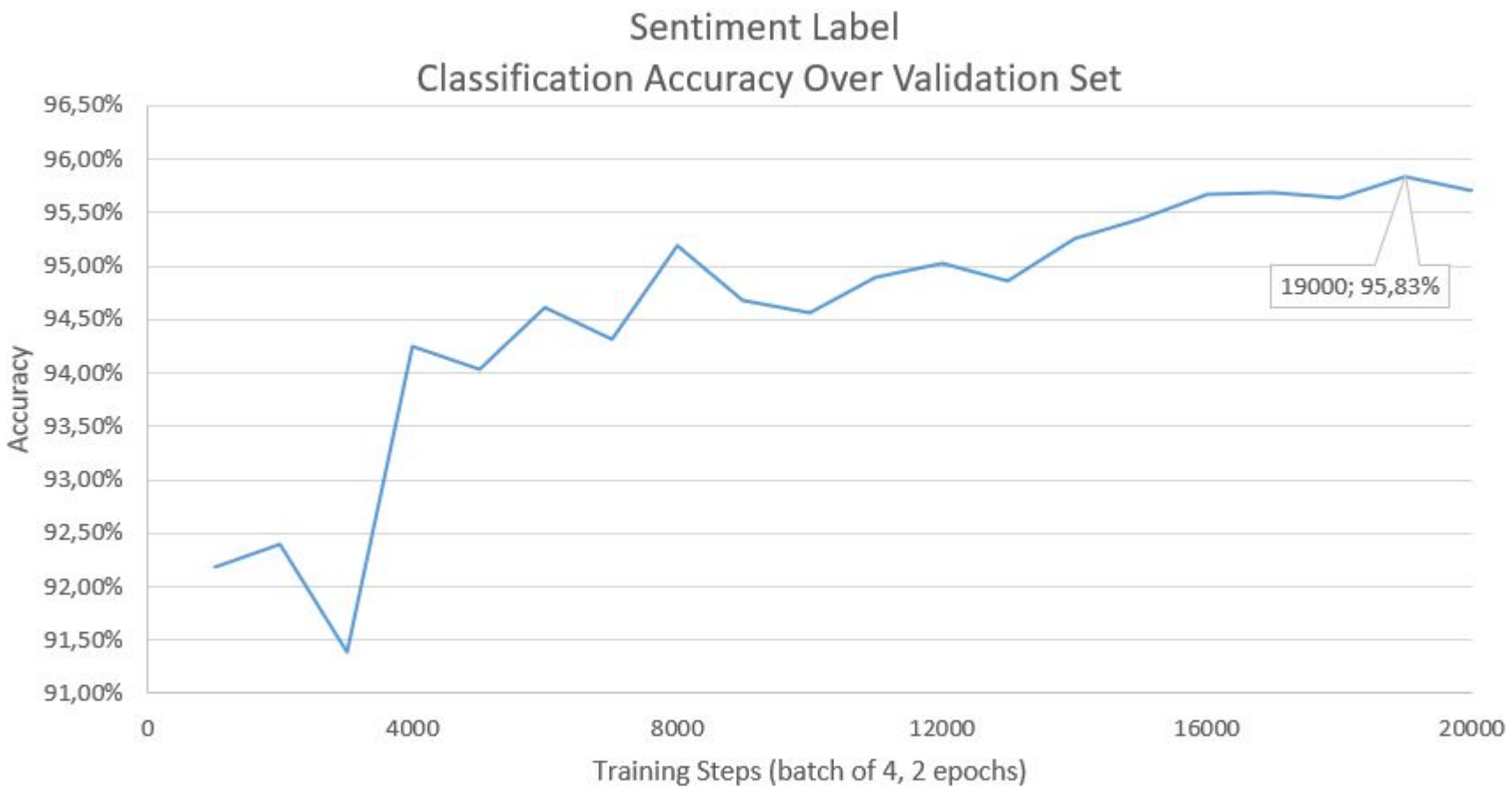
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

S-Bert Training Regime (1)

- Given the siamese-architecture, our working hypothesis is that sbert is best suited for binary classification tasks: given pairs of training data, infer they are similar or not.
- To adapt it for classification, we experimented with the following „hack“:
  - We pair each movie review with a dummy token „<s>“
  - If the review is positive, then set the label as 1, and 0 if review is negative
  - We can therefore train a classical style classifier with a softmax layer sitting on top

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Evaluation



- sbert outperforms bert in our experiment on sentiment

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

S-Bert Training Regime (2)

- Alternatively:
  - We set our loss function as cosine-loss, which tries to maximise the cosine-similarity between the movie review embedding and dummy token embedding
  - After fine-tuning, the embedding on the dummy token branch therefore learns a „positive sentiment“ embedding (as close to 1 as possible, or as far away from negative as possible)
  - Our task is then transformed into a quasi-IR task: given a query (that says „I want positive sentiment“), retrieve from the test dataset all relevant reviews.

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Evaluation

- Because what the model returns is a score of cosine similarity rather than class label, using confusion-matrix to calculate accuracy, precision, recall is not the most appropriate
- We would have to arbitrarily draw a line somewhere that says: above this is positive, below negative
- For this task we report the metric more popular in IR, AvgPrecision (AP score), where:

$$\text{AveP} = \frac{\sum_{k=1}^{n} (P(k) \times \text{rel}(k))}{\text{number of relevant documents}}$$

- We rank the cos-sim scores on test data from highest to lowest, again, rel = 1 if the review is positive, and P is the precision at the cut-off at position k of the list
- NB that precision in an IR setting is defined as:

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Evaluation

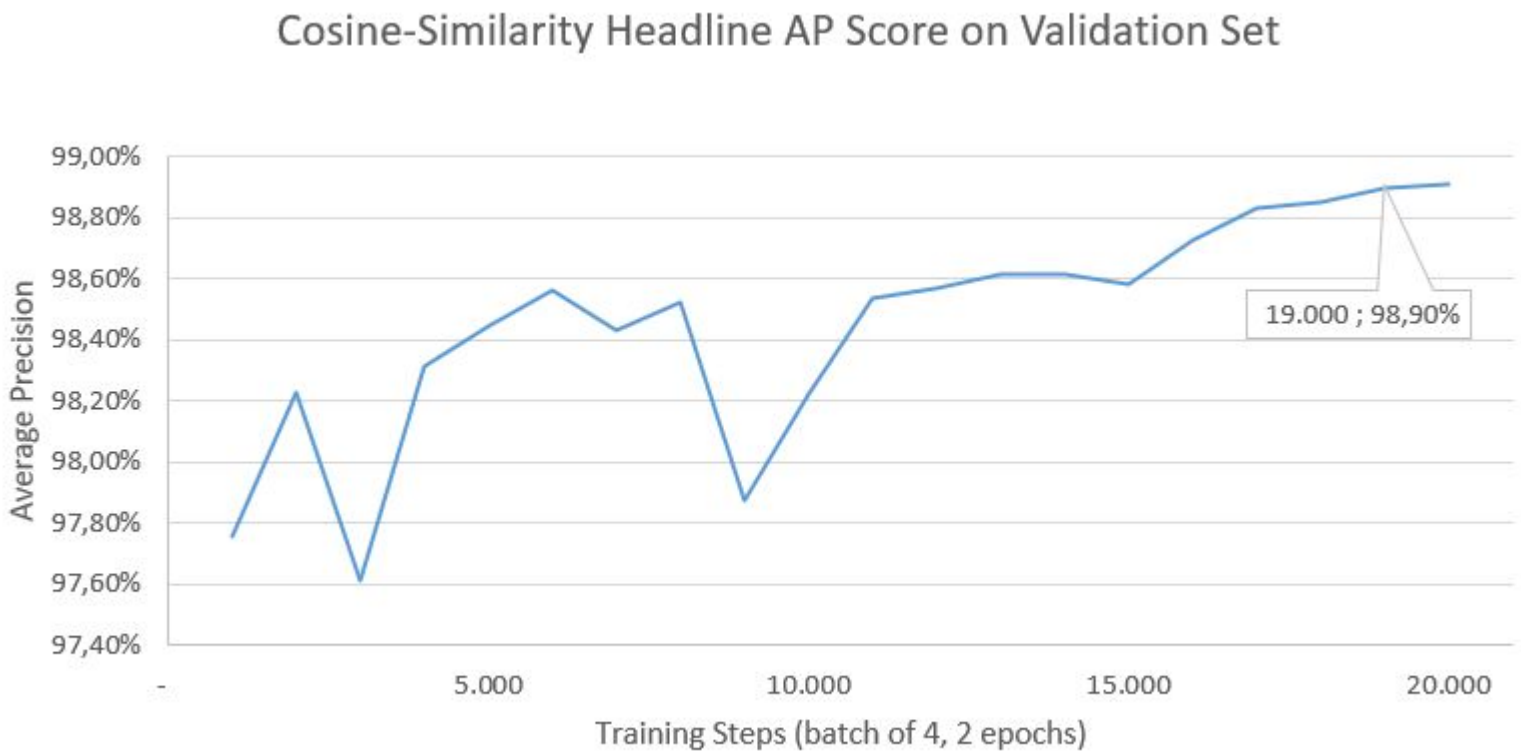
```
class BinaryClassificationEvaluator(SentenceEvaluator):
    def find_best_f1_and_threshold(scores, labels, high_score_more_similar: bool):
        rows = sorted(rows, key=lambda x: x[0], reverse=high_score_more_similar)

        best_f1 = best_precision = best_recall = 0
        threshold = 0
        nextract = 0
        ncorrect = 0
        total_num_duplicates = sum(labels)

        for i in range(len(rows) - 1):
            score, label = rows[i]
            nextract += 1

            if label == 1:
                ncorrect += 1

            if ncorrect > 0:
                precision = ncorrect / nextract
                recall = ncorrect / total_num_duplicates
                f1 = 2 * precision * recall / (precision + recall)
```

Authors' implementation:

- Rank
- Precision

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Evaluation



- significantly outperformed bert

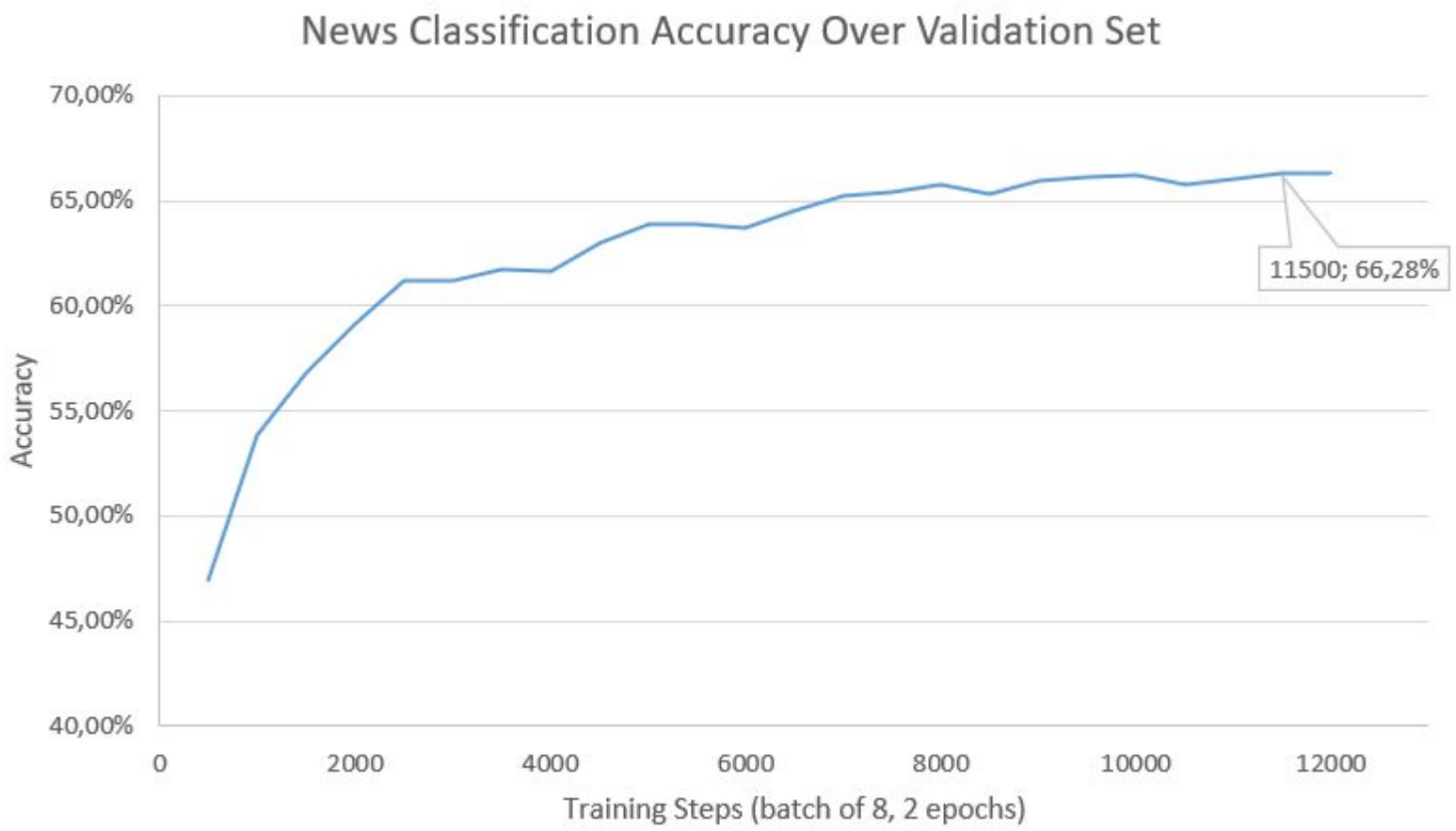
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Multi-class classification?

- As discussed by the authors of the paper, sbert is optimal for NLI tasks, question-answering, or IR with sentence pairs as inputs.
- We experimented on news dataset with the same strategy to convert sbert into a multi-class classifier, again feeding training data paired with dummy, with a softmax layer and cross-entropy loss function. The results as compared with bert is worse.

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Multi-class classification?



LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Discussion

That sbert is not optimal for pairwise classification was also noted by the developers:

- It uses a bi-encoder, i.e. sentences are mapped independently to sentence embeddings.
- For classification tasks, the classifier takes two embeddings and derive a label.
- bert on the other hand, uses a cross-encoder: both sentences are present at input time and bert can compare the two inputs to derive labels. But the disadvantage there is that there is no sentence embedding, which can be used for clustering and semantic search, for example.

- We experimented further with modifying our training data by:
  - For sentiment, passing pairs of positive reviews and negative reviews;
  - For news, passing pairs of sentences with the same label;
  - For news, a zero shot strategy by attaching a k-means block at the end, reframing the task into clustering with k set to number of classes
  - All strategies above delivered significantly worse metrics (for clustering the we reported a silhouette scores <5%)
- Without the either branch the architecture conflates to bert with pooling layer
  - Worse performance will be expected due to this bottleneck downprojection

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

References

- Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.
  - https://github.com/UKPLab/sentence-transformers/tree/master/sentence_transformers
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
  - Bert adapted for sequence classification: https://huggingface.co/docs/transformers/v4.36.1/en/model_doc/bert#transformers.BertForSequenceClassification
- Chicco, D. (2021). Siamese neural networks: An overview. *Artificial neural networks*, 73-94.
- Yang, Y., Yuan, S., Cer, D., Kong, S. Y., Constant, N., Pilar, P., ... & Kurzweil, R. (2018). Learning semantic textual similarity from conversations. *arXiv preprint arXiv:1804.07754*.

LMU LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Thanks for your attention. Questions?