Scalable Machine Learning and Deep Learning - Lab 2

1 Introduction

The goal of this assignment is to train a network to produce sentence embedding, by implementing the proposed method of the Sentence-BERT (S-BERT) paper. In this lab you will get to download and prepare data, fine-tune a pre-trained Transformer network, and finally evaluate on the official Semantic Textual Similarity (STS) benchmark. You are free to use any programming language you see fit, but you are expected to be able to explain parts of your code during the presentation.

The grading for this lab is set to the following progressions:

1. **E:** Implement the S-BERT regression objective (Section 2.1). Evaluate on the official STS benchmark (Section 3.1)
2. **C:** Implement the S-BERT classification objective (Section 2.2), and use it to train a model. Try also first training with the classification object, then fine-tuning using the regression objective (Section 2.3). Compare and evaluate these models on the STS benchmark.
3. **A:** Use your best fine-tuned model and create a small semantic search system (Section 3.2).

Semantic Sentence Embedding

As with most embedding system, the idea is to represent objects as a high-dimensional vectors, so that the distance correlates to a property in which we're interested. In this case, we're going to represent sentences so that the distance of the vector space indicates semantic similarity. Hence, sentences who have a similar meaning should end up close to each other, and sentences who are dissimilar in their meaning should be further apart.

One can define the similarity/distance of two vectors in various different ways. When working with embedding it is very common to use the cosine-similarity, which measure the angle between two vectors. You will however most likely find that most euclidean distance works nearly as good. (I encourage you to try)

2 Training

Sentence-BERT proposes three different training objectives, and all of them start from a pre-trained BERT model. All of them based on the concept of Siamese Networks. However, mainly the classification and regression objective are commonly used today (to my knowledge). Thus only these two are included in the lab. Regardless of the training objective, you are expected to utilize the **Mean-pooling strategy**, described briefly in Section 3 of the S-BERT paper. This means that each sentence embedding is the mean of the outputted *"word"* embedding of the BERT model. Pre-trained BERT models can be found on Huggingface Model Zoo.

2.1 Regression Training Object

The regression loss functions, takes into account the cosine-similarity between two sentence embedding. Hence, allowing us to directly tune the distance of the generated sentence embedding. The official STS training dataset can be found here: STSBenchmark

Since cosine-similarity is always in the range $[-1, 1]$, we can set the most similar sentences to have a similarity of 1, and the most dissimilar to have -1. Sentence pairs with an intermediate label can easily be linearly mapped into this range, to get their target regression label. Table 1 shows an example

Semantic Label	Regression Target
1	-1
2	-0.5
3	0
4	0.5
5	1

Table 1: Examples of linearly mapped regression targets, assuming labels come in the range $[1, 5]$.

of this the mapping from labels to regressions targets, assuming all sentence pairs are labeled in the range $[1, 5]$.

2.2 Classification Training Object

Unfortunately, there is a very limited amount of available STS data. It might therefore be useful to also introduce a different, but conceptually similar task, such as Natural Language Inference (NLI). It is considerably easier to collect NLI data, which has resulted in some decent sized datasets of good quality. The datasets used in the paper can be found in here: SNLI and NultiNLI.

Note that we are still interested in producing sentence embedding, and not solving the NLI task itself. Hence, you should make sure that your model first produces a sentence embedding with the Mean-Pooling-Strategy, and that it is this embedding that is used for the classification.

2.3 Combining both objectives

S-BERT claims that they get even greater performance by first training on NLI and afterwards tuning towers the supervised STS dataset. Your task is therefore to test this hypothesis, and see if you observe a similar effect. Please report the potential gains you see in STS performance when first training on NLI.

3 Evaluation

3.1 STS Benchmark

Evaluation of Semantic Sentence Embedding is usually done by ranking a set of sentence pairs in term of similarity, then comparing the produced ranking with that of humans. The metric used to compare two sets of rankings is the Spearmean Correlation. Although, the paper does this evaluation on multiple different STS datasets (2012-2016), you are only required to do this on the STSBenchmark test set. A straight forward Python implementation of this can be found in Sci-Kit Learn.

3.2 Semantic Search

Finally, we will create something useful of our sentence embedding model. Recall that an embedding is only a high-dimensional vector. This makes searching for nearby embedding, and thus semantically similar sentences, very effective. In particular, one can take a text corpus and pre-compute sentence embedding for all sentences. So that when the user enters a search string, we only need to compute the sentence embedding of the search query.

The task can somewhat be divided up into the following tasks:

1. Download an interesting corpus that you wish to be able to search in. For example perhaps News Headlines?. (You are free to pick any corpus you find interesting).
2. Split the texts of your dataset into sentences.
3. Use your best model to compute sentence embedding for each sentence.
4. Allow the user to entenr a search string, then generate an embedding for this string.
5. Return the K, most similar sentences.