Universität des Saarlandes
Prof. Dr. Dietrich Klakow
Lehrstuhl für Signalverarbeitung
NNTI Winter Term 2021/2022

Language Project

Semantic Text Similarity

Deadline: 04.03.22 23:59

1 Introduction

This is a final project for the course Neural Networks: Theory and Implementation (NNTI). This project will introduce you to Semantic Textual Similarity (STS) task. Semantic textual similarity deals with determining how similar a pair of text documents are. You are expected to make use of concepts that you have learned in the lecture. The project is divided into three tasks, the details of which you can find below.

2 Distribution of points

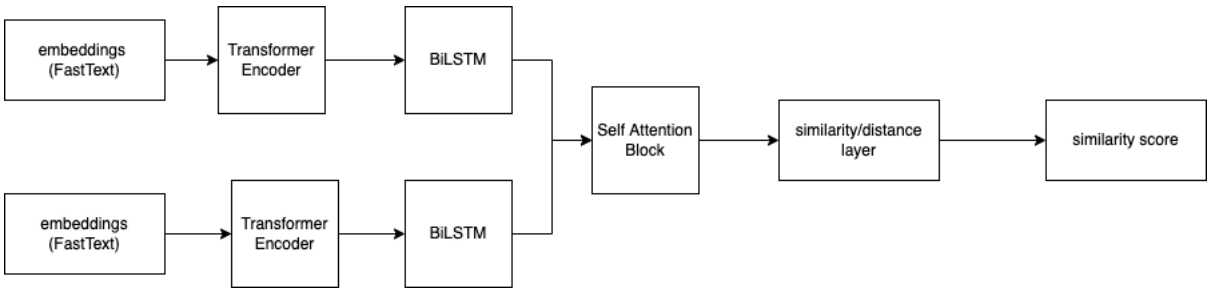
The points in this project are equally distributed among the three tasks. You will be able to score a maximum of 10 points per task, resulting in a total of 30 points in the entire project. How the 10 points are allotted for each task, is mentioned in the guidelines.

3 Task 1: (10 Points)

- In this task you will combine architectures from two papers (Mueller & Thyagarajan, 2016) and (Lin et al., 2017) to solve the STS task. It will form the baseline for the following tasks.
- Follow the instructions in the notebook, complete the code in the accompanying scripts, and run it
- Save the trained models. Load the saved models from disk before evaluating them on test data (do the same for Task 2 and Task 3). That way we can check the accuracy of your results.

4 Task 2: (10 Points)

- In this task, you will enhance your baseline architecture from task 1 by implementing the Transformer Encoder (Vaswani et al., 2017) from scratch and adding to the architecture from task 1. Hence your network architecture should look as follows:



- Note it's strictly not allowed to use ready-made modules `nn.TransformerEncoderLayer`, `nn.TransformerEncoder` and `nn.MultiheadAttention` from PyTorch. You should implement all the building blocks from scratch (basic PyTorch API).
- Compare and analyze the results with Task 1 and discuss that in the report. Additionally, you may visualize the attention scores on sequences across sentence pairs. You should add additional visualizations, analysis and discuss the potential factors behind your results.
- Ensure a modular implementation to have the number of encoder layers configurable via a hyperparameter.
- You are allowed to consult online resources, however, don't forget to cite the resources that you find the most helpful.
- It is highly recommended to follow the structure of scripts and naming conventions from Task 1.

5 Task 3: Challenge Task (10 Points)

In this third and final task of the project, you are expected to

- Read multiple resources about the state-of-the-art work related to Semantic Textual Similarity.
- Try to come up with methodologies that would possibly improve your existing results.
- Improve on the accuracy scores from Task 1 and 2.
- Note that the improvement can either be in terms of run time performance of the architectures from Task1 and Task2, for instance as proposed by (Mehta et al., 2021) or on the performance metrics - one potential SoTA approach is proposed in (Reimers & Gurevych, 2019).
- Most importantly in this part you can implement a paper of your choice or your own idea which potentially improves the performance. However, it is not mandatory to get high scores, the main focus of our grading will be on your observations and analysis of the results.
- Include the analysis of your observations in the report.

Note: The task here should be a change in the model architecture, data representation, different approach, or some other similar considerable change in your process pipeline. Please note that although you should consider fine-tuning the model hyperparameters manually, just doing that does not count as a change here.

6 General Guidelines

- In case you face any difficulties with the project, please raise a question on Piazza. Alternatively you can get in touch with one of the tutors in charge (Shahrukh Khan or Daniyar Kazbek) using MS-Teams (but Piazza is more preferable).
- For Task1 and Task2 you are not allowed to use ready-made libraries like Hugging Face Transformers.
- For Task3 if you decide to use any pretrained model make sure to finetune it using a custom (PyTorch) training loop.
- You are allowed to use convenience methods for data loading & preprocessing from packages like scipy, numpy, pandas, sklearn, NLTK. If you use other packages, provide a reference and justify it.
- Plagiarism will be penalized and can eventually lead to disqualification from the project and the course. Most importantly, we will check for plagiarism within groups. If we see any clear indication of plagiarism among groups, both groups will be awarded 0 for the whole project. Discussion with groups is allowed (only in terms of concepts but not directly with code).
- Cite any resources that you found to be helpful.
- You are expected to provide a separate notebook for each task. However, the notebook should only contain runtime code. Functions or classes you write should be in separate python scripts (.py) that are imported into the notebook of that task, similar to the skeleton structure of task 1. All submitted code should be runnable.
- Your code should be sufficiently commented so that we can grade it easily. Not providing proper documentation can lead to your code not being graded.
- Write a well-documented academic report. The report needs to be 6 (at most) pages long following the NeurIPS format. You can have a look at Latex versions or in other formats. We expect a .pdf file from you. The way how you divide it is up to you but we roughly expect to have an introduction, methodology, results, and conclusion sections. Of course, you will have to cite every source that you use.
- The main focus of our grading will be your observations and analysis of the results. Even though you might obtain bad results make comments on what could have gone wrong.

7 Grading Breakdown

a) Task 1
   - Implementation (10 points)

b) Task 2
   - Implementation (5 points)
   - Analysis, interpretations, report (5 points)

c) Task 3
   - Implementation (5 points)
   - Analysis, interpretations, report (5 points)

8 Submission instructions

- Use the NeurIPS 2021 Template when writing your report. Other conference formats or .doc[x] won't be accepted.
- You are required to submit the final project as a team of two students.
- You should submit a detailed report, implementations, and saved models in a zip file. Link to the repository should also suffice.
- Make sure to write the MS Teams username, matriculation number, and the name of each member of your team on your submission.
- If your zip file size exceeds the limit of the allowed submission limit on MS-Teams then do the following. Create a zip file of your your project with a downloadable link to the zip file which is not password protected. You may use google drive, dropbox, etc for this. Compute the md5sum of this file and send us the downloadable link and the md5sum of the file along with your submission on MS-Teams before the deadline. We will generate the checksum for the zip file and match it with the checksum you provide us.
- If you have any trouble with the submission, contact the tutors before the deadline.

References

Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017). A structured self-attentive sentence embedding.

Mehta, S., Ghazvininejad, M., Iyer, S., Zettlemoyer, L., & Hajishirzi, H. (2021). Delight: Deep and light-weight transformer.

Mueller, J., & Thyagarajan, A. (2016). Siamese recurrent architectures for learning sentence similarity. *Proceedings of the AAAI conference on artificial intelligence*, *30*(1).

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need.