Deep Learning for Natural Language Processing

SS21 – project sheet – 20 points

Submission start via LernraumPLUS: 24.07.2021, 12 pm
Submission deadline via LernraumPLUS: 01.09.2021, 10 pm

In this project you will investigate pre-trained language models (transformers)

`https://towardsdatascience.com/pre-trained-language-models-simplified-b8ec80c62217`[1]
gives a very broad overview

Everybody works on a individual NLP-task-transformer-combination. The project includes deep self-informing about the task (NLP area, the task itself, challenges and current approaches to solving the task), providing a pre-trained-language-model-approach to solve the task and writing a short paper for this task.

Pretrained languages are usually very big, and even the fine-tuning takes lots of time. However, each transformer-type has different size-variants and it's totally ok when you take the smallest size (faster trainable) - even if the results with the small-size-variants are possible not so good in comparison to the largest model. If you have the computational resources you can use larger variants, of course, but we will not grade the computational capacities ;).

We want consider following transformers

1. **BERT** (size-variants (incl. case-sensitive or not): `bert-base-cased`, `bert-base-uncased`, `bert-large-uncased`, `bert-large-cased` `bert-large-uncased`, `bert-xlarge-v2`, `bert-xxlarge-v2`)
2. **D**istil**BERT** (size-variants (incl. case-sensitive or not): `distilbert-base-cased`, `distilbert-base-uncase` `distilbert-large-uncased`, `distilbert-large-cased`
3. **A**L**BERT** (size-variants: `albert-base-v2`, `albert-large-v2`, `albert-xlarge-v2`, `albert-xxlarge-v2`)
4. **R**o**BERT**a (size-variants: `albert-base`, `albert-large`)
5. **T5** (size-variants: `t5-small`, `t5-base`, `t5-large`) – *this is a special transformer genertin a textual output rather then a classification for example. Pre- and postprocessing can be more difficult then the other transformers - therefore, we will consider this in our grading*

And the tasks now Not all transformers are sensible appropriate for every task. Therefore, we give already some hints that you should consider:

☺ the transformer is appropriate
☹ it's not recommend, but it's at least a try
≈ not appropriate

Task	BERT	DBERT	ABERT	RBERT	T5
Key Point Analysis Shared Task - Track 1 (Key-Point Matching)	☹	☺	☺	☺	≈
Key Point Analysis Shared Task - Track 2 (Key Points Generation WITHOUT Matching)	☺	☺	☺	☺	☹
Frame Identification - data	☹	☹	☹	☺	☺
SemEval-2020 Task 7: Assessing Humor in Edited News Headlines - Subtask 1 (Regression) [data]	☺	☺	☺	☺	≈
SemEval-2020 Task 7: Assessing Humor in Edited News Headlines - Subtask 2 (Funnier) [data]	☺	☺	☺	☺	≈
Math Question Answering	☹	☹	☹	☺	☹
Same Side Classification Task	☺	☺	☺	☺	☹
Claim stance classification: IBM Debater	☺	☺	☺	☺	≈
Evidence Quality: IBM Debater	☺	☺	☺	☺	≈

[1] *SOTA* means **S**tate **o**f **t**he **a**rt (that refers to the (best) current solutions/ approaches to solve a particular task)

Sheet 5 – Task Choice (1 points)

Task until 21.07.2020, 10 pm: Please choose two topics of the table: the first one should be your favourite topic, the second one your second choice. Additionally, you can specify your transformer favourites (for each task), but this is not mandatory. We recommend to inform yourself beforehand.
Please send the preference list to `pheinisch@techfak.uni-bielefeld.de`. I will try to allocate the topics in such a way that everybody is satisfied with the received topic. Every topic-transformer-combination is assigned to max. 1 person.
Pay attention to your exercise points. It is required to have at least 50% of the exercise points. The exercise points do not have an influence on the final grade, only the quality of the project submission!

Sheet 5 – Task Coding (9 points)

Provide a **own programmed** code which solves your task. Your approach should use the assigned pre-trained language model (at least as a part of the final neural net).
Your code should be separated into five files:

- `train-evaluate-main.py`: a files which trains, evaluates and saves the fine-tuned model in a `if __name__ == "__main__"`-block
- `preprocssing.py`: a file which loads the raw corpus data and pre-process it – the file should deliver the specific tensors which can directly used by the learning model
- `model.py`: a file which defines the neural net
- `test.py`: a file which fetches a task specific example and prints the task specific output, relaying on the loaded fine-tuned learning model. For example: if you have the *Frame Identification* task, the file should feed with a English text like "STUDENTS SHOULD DESIRE GOOD GRADES IN UNIVERSITY BECAUSE THIS INCREASES THE CHANCE TO GET A JOB WITH A HIGH SALARY" and should print the regarding frame. Hence, a optimal trained learning models would output: "ECONOMICS".
- `ReadMe.txt` (or a Markdown-file, hence `ReadMe.md`): a file which inform about the correct use of the code, especially the use of `test.py`. The file includes a list of list of required python libraries, too.

Besides, your submission should contain a saved fine-tuned learning model. If the fine-tuned-model-file[s] exceed the maximum submission size in *LernraumPLUS*, try to compress them. If this is still not sufficient, please provide the file[s] somewhere else and give a link.
You should program with Python Python 3.8. We recommend to rely on tensorflow (all listed transformers are available in a tensorflow-version).
All of your code must be compilable! Proper maintainable code will have a good effect on your grade, too. Apt maintainable code includes sensible comments, carefully selected variable names and uses methods (primarily if you use a code block more than one time).

Sheet 5 – Task Paper (10 points)

Submit a self-written mini-paper about your assigned task, too. Your paper should contain the following sections:

- **Introduction**: introduces into the task, describes it, motivates (why we should solve the task) shows challenges (why is the task not a trivial one?) and gives an overview about the paper
- **Related work**: the chapter summarizes an already existing approach that tries to solve the task and differs from your approach. You should select *one* approach (scientific paper[a]) on your own.
- **Method**: describes your method and model architecture and introduces your selected language model (give some reasons [advantages] why you go with your assigned language model)
- **Evaluation**: provides the test results and compares them to the related work in your mini-paper (in case of a shared task with a results-table which lists the results of existing approaches, consider the table, too)
- **Conclusion**: draws a conclusion about your work and in case of worse results tries to explain them

Your mini-paper should have a length of 4 pages (not more!). Your paper should end with bibliography with at least two references. You can place this bibliography on an extra page, too. Informative pictures are an advantage. The standard font size should be 11pt. Please write in English.
You can use the provided LaTeX- template. However, you're not forced to work with LaTeX.

[a] If you have a shared task, you can find easily existing approaches by looking on the website of the shared task (or google the shared task). If you don't have a shared task, `https://scholar.google.com/` may help

Please don't hesitate to ask when you have questions. An optional offer is one virtual meeting with me (Philipp Heinisch – `pheinisch@techfak.uni-bielefeld.de`) after the task assignments. In the one virtual meeting, you can clarify your questions if there are any. The session can be in German, too.