Week 3: Transformers, BERT and Transfer Learning with Language Models

All the referenced files are available on LearnIt in the archive `assignment-2.tar.gz`. The goal of this assignment is to:

- Learn how to analyze the tokenizer of a language model
- Be able to train, use, and evaluate a transformer-based language model for NLP tasks
- Evaluate the language-transfer abilities of multilingual language models

Using containers on the HPC

For this course, we will be using the datastes library, and the transformers library, which makes use of pytorch. To simplify installing the relevant packages, our HPC admin has prepared a container containing the neccesary packages. To use it you need to include the following in your sbatch file:

```
module load singularity

singularity exec --nv /opt/itu/containers/pytorchtransformers/pytorch-24.07-py3-transformers.sif
python my_python_script.py
```

It's also possible to install more packages within the container using pip:

```
singularity exec --nv /opt/itu/containers/pytorchtransformers/pytorch-24.07-py3-transformers.sif
pip install $python_package
```

Installed python packages will persist between runs (they get installed into `.local/lib/python3.10`, or your venv location, if using that). The documentation for using containers on the HPC is here: `http://hpc.itu.dk/software/containers/PremadeContainers/`

1 Subword tokenization

Bert-based models are usually trained with a fixed vocabulary that is obtained through training a subword tokenizer (e.g. Schuster and Nakajima, 2012). For this assignment, we are going to inspect the behaviour of the tokenizer of the most commonly used multilingual model mBERT.[1]

The goal of the word segmentation step is to go in-between representations of characters and words, so that we can maintain high efficiency and also benefit from sharing while minimizing the unknown word problem. It is commonly believed that the goal of word segmentation is to match morphemes (the smallest meaningful sequence of characters in language). The morphemes of 'unbeatable' are typically annotated as: 'un'-'beat'-'able'.

In the transformers library you can use a subword tokenizer as follows:

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("bert-base-multilingual-cased")
print(tokenizer.tokenize("this is an example input"))

> ['this', 'is', 'an', 'example', 'input']
```

[1] `https://github.com/google-research/bert/blob/master/multilingual.md`

(a) Compare the tokenizations of the mBERT tokenizer of texts from two different language(-varietie)s you are able to understand/read. Use English, the dominant language in mBERT, with a lower-resource language variety (for example Danish). If you only know 1 language, try to use a different variety of the language (for example for English, use social media abbreviations or typos, e.g.: c u tmrw). You can collect data from any source, or make up your own sentences.

(b) Now test the tokenizer of a language model that is trained for your target language. You can find language models on `https://huggingface.co/search/full-text?type=model`. Can you observe any differences in the results (in amount/length of subwords)? Do the results match your intuition of separating mostly short meaning-carrying subwords?

(c) Think of two example inputs where the tokenizer might struggle to find a meaningful segmentation (for example by introducing typos). Why are these cases difficult?, did the tokenizer do something sensible?

2 Cross-domain transfer

In this assignment we are going to fine-tune BERT models. Note that you need ≈ 8GB of GPU-ram for this assignment (or a lot of patience). There are GPU's available on the HPC (`hpc.itu.dk`, introductory slides can be found on LearnIt).

Hint: You can use a gpu node on the HPC cluster interactively for debugging (e.g. install packages) with a command like:

```
srun --job-name "interactive" --cpus-per-task 2 --time 1:00:00 --gres gpu --mem 12G --partition brown --pty bash
```

We will use the sentiment analysis datasets from assignment 1, and see if we can achieve better cross-domain performance by using a language model trained on social media data. We will use `cardiffnlp/twitter-xlm-roberta-base` (Barbieri et al., 2022), which is a multilingual language model that is re-trained on social media for language modeling.

You can use the code in `bert-classification.py` for the following assignments (note that it gives a warning about parameter renaming, this can be ignored).

(a) Train a sentiment analysis model with BERT (`bert-base-cased`) on the English SST data. Evaluate it on the SST data as well as on the English Twitter data from SemEval2013. Is there a similar performance drop as in assignment 1?

(b) Inspect the code and try to understand the steps of the inference and the training procedure. What is the shape of the `output_scores` variable of the `forward` function?, what do the dimensions represent?

(c) Now train a sentiment model with the twitter embeddings (`cardiffnlp/twitter-xlm-roberta-base`) with the SST train data. Does it transfer better to the English Twitter data compared to the mBERT model?

3 Cross-lingual transfer

In this assignment we will look at cross-lingual transfer. We will compare the cross-lingual performance of a monolingual language model to the performance of a multilingual language model. We will use the SST English data for sentiment analysis for training, and evaluate the models on Danish data (`https://github.com/alexandrainst/danlp/blob/master/docs/docs/datasets.md#twitter-sentiment-twitsent`).

(a) Train an English BERT model (`bert-base-cased` on huggingface) on the reviews data from SST, and evaluate it on the SST data as well as the Danish Twitter data. Is there a performance drop when going to the Danish data? How does the performance on the Danish data compare to the majority baseline?

(b) Train an mBERT model (`bert-base-multilingual-cased` on huggingface) model on the reviews data from SST, and evaluate it on the Danish development split, what is the performance? How does it compare to the English BERT model?

Week 4: Autoregressive language models

The goal of this assignment is to:

- Learn how to do inference with autoregressive language models
- Evaluate how to extract useful context information from a large text corpus
- Compare the effect of fine-tuning versus RAG
- Be aware of difficulties in evaluation of autoregressive language models
- Get experience with re-training language models

We will make use of the FLAN-T5-base language model; note that it can probably run inference on your own machine (needs ≈4gb to run), but it will be much faster with a GPU (a speedup of ≈ 10-50x). For the fine-tuning assignment we would recommend using a GPU. Note that you can use the HPC (`www.hpc.itu.dk`) for this.

4 Question answering with FLAN-T5

You can use the code in `qa.py` as a starting point. For evaluation, it checks whether should be noted that the evaluation metric is a custom metric "designed" by Rob, it checks whether at least half of the gold words in the predicted output.

1. Add 5 common-knowledge questions and answers in the corresponding python lists.
2. What is the performance of FLAN-t5 base on your questions?
3. What are possible pitfals of the evaluation metric?
4. If the model has made some errors: why is this the case?, did it misunderstand the question, or does it not know the answer? (if the model made no errors, think of 2 more difficult questions)
5. Experiment with at least 2 different prefixes and postfixes; do they improve performance?

5 Domain Adaptation through Retrieval Augmented Generation

Now we will evaluate the FLAN-T5 model on the Star Wars domain. For this, I have scraped 66 Star Wars trivia questions from `https://parade.com/1161189/alexandra-hurtado/star-wars-trivia/` they have been pre-processed and are available in `questions.txt` and `answers.txt`.

1. What is the performance of the FLAN-T5 model out-of-the-box?
2. Can you improve performance with your pre- and post- fixes? Why?
3. We also provide you with the raw text from Wookieepedia, this is a fandom wiki with information about the Star Wars universe written in English. It has been scraped using the procedure described on `https://robvanderg.github.io/datasets/wikia/`, and is available in `starwarsfandomcom-20200223.txt.cleaned.tok.un`
   Use the words from the questions to find the 5 sentences with the highest word overlap (the raw sentences with the largest coverage of words with respect to the question). Add these sentences as a prefix, separated with newlines. Does performance increase?

4. **Bonus**: experiment with better variants of data selection, what is the highest score you can obtain? Note that ChatGPT achieved a score of 51; feel free to also use larger language models, for example `google/flan-t5-large`.

6 Domain adaptation through fine tuning

Fine tune the FLAN-T5 model on the raw wookieepedia.txt data. You can read in the documentation (`https://huggingface.co/docs/transformers/model_doc/t5#training` about how T5 can be finetuned. Since no PyTorch implementation is available, we provide `run_t5_mlm_torch.py`, which can do language modeling on raw data with a T5 model.

Note: I have also finetuned the language model, it is available on: `https://huggingface.co/robvanderg/flan-t5-base-starwars`. We would recommend you try to finetune the language model yourself and play around with the parameters while inspecting the loss. This is a very practical and useful skill to acquire.

1. What is the performance of the fine-tuned FLAN-T5? What can we now conclude about data selection for prefixing versus fine tuning (within this setup)?

2. Combine the best prefixing method you found in assignment 5 with the fine-tuned model. Did performance improve further?

3. **Bonus** Experiment with different parameters for all settings (LM, pre-/post-fix, re-training). How many instances can you manage to get correct in total?

Hint: Note that the transformers library automatically downloads the language models to `/home/user/.cache/huggingface/transformers/`. You can empty this folder after completing the assignments.

References

F. Barbieri, L. Espinosa Anke, and J. Camacho-Collados. XLM-T: Multilingual language models in twitter for sentiment analysis and beyond. In *Proceedings of the Language Resources and Evaluation Conference*, pages 258–266, Marseille, France, June 2022. European Language Resources Association. URL `http://www.lrec-conf.org/proceedings/lrec2022/pdf/2022.lrec-1.27.pdf`.

M. Schuster and K. Nakajima. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE, 2012.