Exercises on Natural Language Processing with Transformers

Ion Androutsopoulos, 2023–24

Submit as a group of 2–3 members (unless specified otherwise in the lectures) a report (max. 10 pages, PDF format) for exercises 1 and 2. Include in your report all the required information, especially experimental results. Do not include code in the report, but include a link to a Colab notebook with your code. Make sure to divide fairly the work of your group to its members and describe in your report the contribution of each member. The contribution of each member will also be checked during the oral examination of your submission. For delayed submissions, one point will be subtracted per day of delay.

1. Repeat Exercise 2 of Part 5 (sentiment classifier), by fine-tuning a pre-trained BERT model.[1] Tune the hyper-parameters (e.g., sizes of any task-specific layers on top of BERT, number of BERT encoder blocks to keep frozen) on the development subset of your dataset. Monitor the performance of your models on the development subset during training to decide how many epochs to use. If the texts of your experiments exceed BERT's maximum length limit, you may want to truncate them at the maximum allowed length of BERT or use a BERT-like model that can handle longer texts (e.g., Longformer).[2] Include experimental results of a baseline majority classifier, as well as experimental results of your best classifiers from exercise 15 of Part 2, exercise 9 of Part 3, exercise 1 of Part 4, exercise 2 of Part 5, now treated as additional baselines. Otherwise, the contents of your report should be as in exercise 2 of Part 5, but now with information and results for the experiments of this exercise. You may optionally include (for extra bonus) indicative experimental results on a small subset of the test set (e.g., 10 test examples) obtained by prompting an LLM (e.g., Chat-GPT), using appropriate instructions and possibly including few-shot examples (demonstrators).[3]

2. Repeat Exercise 3 of Part 5 (POS tagger), by fine-tuning a pre-trained BERT model. Tune the hyper-parameters on the development subset of your dataset. Monitor the performance of your models on the development subset during training to decide how many epochs to use. If the sentences of your experiments exceed BERT's maximum length limit, you may want to truncate them at the maximum allowed length of BERT or use a BERT-like model that can handle longer texts (e.g., Longformer). Include experimental results of a baseline that tags each word with the most frequent tag it had in the training data; for words that were not encountered in the training data, the baseline should return the most frequent tag (over all words) of the training data. Also include experimental results of your best method from exercise 10 of Part 3, exercise 2 of Part 4, exercise 3 of Part 5, now treated as additional baselines. Otherwise, the contents of your report should be as in exercise 3 of Part 5, but now with information and results for the experiments of this exercise. You may optionally include (for extra bonus) indicative experimental results on a small subset of the test set (e.g., 10 test examples) obtained by prompting an LLM (e.g., Chat-GPT), using appropriate instructions and possibly including few-shot examples (demonstrators).

---

[1] You can use, for example, https://huggingface.co/transformers/.

[2] See https://huggingface.co/docs/transformers/model_doc/longformer.

[3] See, for example, https://chat.openai.com/auth/login, https://chat.lmsys.org/, https://huggingface.co/spaces/ysharma/Explore_llamav2_with_TGI.

3. (a) We were given a BERT model pre-trained on a generic English corpus and we want to use it to build a Machine Reading Comprehension (MRC) system. The MRC system will be given a question and a paragraph (as shown in the figure) and will aim to predict the spans (sequences of tokens) of the paragraph that answer the question. The first token of each answer span should be classified as B (begin), the other tokens of the answer span as I (inside), and all the other tokens of the paragraph as O (outside). Let h_i be BERT's top-level representation of the i-th token of the paragraph and let $p_i \in [0,1]^3$ be the probability distribution over the three classes (B, I, O) produced by the MRC model for the same token. We add a task-specific dense layer on top of BERT to obtain the p_i distribution for each token of the paragraph from the corresponding h_i. **Write a formula showing how p_i is obtained from h_i**, assuming $h_i \in \mathbb{R}^{128}$. Also **write down the dimensions of all the matrices and vectors** used in the formula.

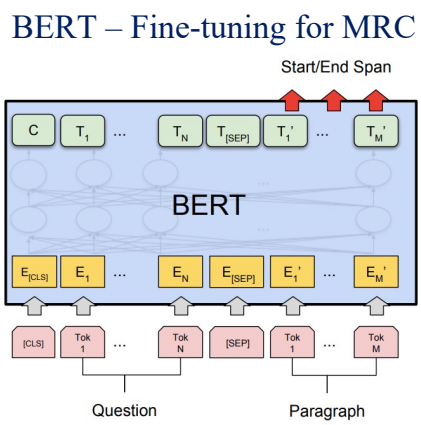

Figure from Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018.

Formula: $p_i = \text{softmax}(Wh_i + b)$, where $p_{i,j} = \frac{[\exp(Wh_i+b)]_j}{\sum_{k=1}^{3}[\exp(Wh_i+b)]_k}$ and $j \in \{1,2,3\}$.

Dimensions: $W \in \mathbb{R}^{3\times128}, b \in \mathbb{R}^3$

(b) Write a <u>detailed formula</u> showing how you would compute the overall loss (L) for an input training instance when training the model (jointly fine-tuning BERT and training the task specific dense layer on top). Assume for simplicity that in all the training instances, the question is N_Q tokens long and the paragraph is N_P tokens long. Call t_i the correct (gold) output probability distribution for the i-th token of the paragraph. **Do not assume that every t_i is 1-hot**. For example, t_i may be a gold per-token probability distribution over the classes, based on the opinion of multiple human annotators; we may have three annotators, two of them may have said that the i-th token is a B, and the third annotator may have said it is an O, in which case the gold distribution for the token over B, I, O is 2/3, 0, 1/3.

Loss: $L = -\sum_{i=1}^{N_p}\sum_{j=1}^{3} t_{i,j} \log p_{i,j}$

(c) Now assume that every t_i is 1-hot. Show **how the formula of the previous sub-question can be simplified**. Clearly explain the steps of the simplification.

Solution:

Now for every token (at position i) of the paragraph, $t_{i,j} = 1$ if the correct class of the token is the j-th one, and $t_{i,j} = 0$ otherwise. Let $r(i)$ be the (index of the) correct class of the i-th token. Then the loss becomes:

$$L = -\sum_{i=1}^{N_p} t_{i,r(i)} \log p_{i,r(i)} = \sum_{i=1}^{N_p} \log p_{i,r(i)}$$

i.e., we maximize the log-likelihood of the correct classes of the paragraph's tokens.

(d) The MRC system of questions (a)–(c) will actually be used in the **biomedical domain**. We have **3k annotated training instances** (question-paragraph pairs with gold answer spans) **from the biomedical domain**, along with **500k additional plain text paragraphs of biomedical text** (without any questions and answer spans). The BERT model we have was pre-trained (using a masked language modeling and a next sentence prediction loss) on millions of plain text inputs, but from a generic corpus that contains very few biomedical documents. **How could we use the additional 500k plain text biomedical paragraphs to improve the performance of our MRC system?** You do not need to provide any formulae for this sub-question, but **your answer must be sufficiently detailed for an experienced colleague** (e.g., another student of the course) **to understand and implement your idea(s).**

One possible answer:

We can take the BERT model that is already pre-trained on millions of plain text inputs and further pre-train it (with masked language modeling loss and next sentence prediction loss) on the 500k additional biomedical plain text paragraphs to tailor it to the biomedical domain. We could then fine tune the pre-trained model on the 3k annotated MRC training instances of the biomedical domain, as in questions (a)–(c).