Assignment 4: Text Generation and Question Answering with Transformers

Deadline: Tuesday November 15, 2022 at 9:00pm
Late Penalty: There is a penalty-free grace period of one hour past the deadline. Any work that is submitted between 1 hour and 24 hours past the deadline will receive a 20% grade deduction. No other late work is accepted.

The goal of the fourth assignment is to gain familiarity with the nature of auto-regressive language generation using Transformers, through the lens of the Question-Answering (QA) task. You will also fine-tune a model from Huggingface pre-trained models, with their easy to use software, to do QA on a part of the well-known SQuAD dataset. You will also explore of the generated probabilities, and zero-shot prompting of generation.

This assignment must be done individually. The specific learning objectives in this assignment are:

1. Become familiar with some forms of the Question-Answering problem.
2. Download and use pre-trained QA models with Huggingface Transformers.
3. Fine-tune a QA model using a Huggingface-based Transformer.
4. Learn about the generation (decoding) process and viewing the generation probability tree, as well as the effect key generation-controlling hyperparameters.
5. Learning about zero-shot generation.

What To Submit

You should hand in the following files to this assignment on Quercus:

- A PDF file `assignment4.pdf` containing answers to the written questions in this assignment. You should number your answer to each question in the form `Question X.Y.Z`, where X is the section of this assignment, Y is the subsection, and Z is the numbered element in the question. **You should include the specific written question itself** and then provide your answer.
- Code files as specified in individual questions.

1 The Problem Setting of QA (3 points)

There are many different ways that the QA problem is posed. Here we focus on the setting specified by the SQuAD [2] dataset. The paragraphs below show three QA problems sampled from the SQuAD dataset, with minor modifications. For each question listed below, provide the answer (using your own brain, not the computer) based on the context given and the question. For each answer that you give, also write a short reason that justifies your answer. [3 points]

1. **Item ID 5709a923ed30961900e843d0**
   Context: Unlike the Spanish milled dollar the U.S. dollar is based upon a decimal system of values. In addition to the dollar the coinage act officially established monetary units of

mill or one-thousandth of a dollar, cent or one-hundredth of a dollar, dime or one-tenth of a dollar, and eagle or ten dollars, with prescribed weights and composition of gold, silver, or copper for each. It was proposed in the mid-1800s that one hundred dollars be known as a union, but no union coins were ever struck and only patterns for the $50 half union exist.
Question: What is the US dollar based upon?

2. **Item ID 56fa3d788f12f319006300ff**
   Context: Age, diameter, height, radial growth, geographical location, site and growing conditions, silvicultural treatment, and seed source, all to some degree influence wood density. Variation is to be expected. Within an individual tree, the variation in wood density is often as great as or even greater than that between different trees (Timell 1986). Variation of specific gravity within the bole of a tree can occur in either the horizontal or vertical direction.
   Question: Which part of a tree can have vertical or horizontal variation in its specific gravity?

3. **Item ID 56e102b7cd28a01900c6742b**
   Context: Canonical jurisprudential theory generally follows the principles of Aristotelian-Thomistic legal philosophy. While the term "law" is never explicitly defined in the Code, the Catechism of the Catholic Church cites Aquinas in defining law as "...an ordinance of reason for the common good, promulgated by the one who is in charge of the community" and reformulates it as "...a rule of conduct enacted by competent authority for the sake of the common good."
   Question: What school of thought serves as a model for canon theory?

2 Performing QA Using a Pre-Trained Model (6 points)

In this section you will use a pre-trained, online model to answer the above three questions, so no coding is involved. You will use the deepset/roberta-base-squad2 pre-trained model from the huggingface model hub. If you follow that link you will see the 'Hosted inference API' on the right hand side, which you should use to answer these questions:

1. For each of the three question answering problems in Section 1, copy the Question and the Context into the model interface and obtain the resulting generation. Report the results, and comment on how the model outputs compare to your answers from Section 1. Create your own context and question to query this same model, and report the answer. Is it correct? [2 points]

2. Read the pipeline tutorial from Huggingface to see the most straightforward way to use models from the Huggingface model hub. Review all of the tasks that are supported by the pipeline framework, and list those that take in text as input. For each of those, give a one line description of what it does. [2 points]

3. The model page for deepset/roberta-base-squad2 has a section labelled "In Transformers" that uses the pipeline method, and it can do exactly the same thing that you just did in part 1 of this Section. Review and copy that code, make sure you can run the three example QA problems. Find a different model that can perform inference at least 25% faster (in wall clock time) than `roberta-base-squad2` that can be used in that same pipeline, and see if it gives the same answers - report which model you chose, its inference time (and the time for

roberta-base-squad2), and whether the answers were comparable to roberta-base-squad2 or not. [2 points]

3 QA by fine-tuning a model on SQuAD (12 points)

Read this tutorial to learn how to fine-tune an existing model on a specific question-answering dataset.

1. Answer the following questions:
   (a) If you do not wish to upload the newly trained model to the Hugging face hub, is it necessary to execute the `notebook_login` line of the notebook? [0.5 points]
   (b) Give a single line of code that will print the 5th data point in the training split of the SQuAD dataset? [0.5 points]
   (c) Some input examples may exceed the maximum input length of the model. The code in the tutorial truncates inputs that are greater than a maximum length, as part of the `tokenizer`. It can split the input into two *segments*, one segment which fits and another segment that is what remains. When calling the tokenizer, which argument causes the tokenizer to return both the segments after truncation? If we want to have the two segments have some overlap (just in case the text containing the answer is truncated), which argument should be used to control the extent of overlapping? [1 point]
   (d) How are input examples that are not the same length handled in this code? [1 point]
   (e) In the function `prepare_train_features()`, are the `start_positions` and `end_positions` referring to the token indices or the character indices? [0.5 points]
   (f) Which gradient descent optimizer is used in this code? If we want to change the optimizer, how is this done? (Hint: look at the `TrainingArguments` and the `Trainer`) [1 point]
   (g) What is "checkpoint saving", and why is it done? How often is checkpoint saving done in the model training in this code? [1.5 points]
   (h) How is the performance of the model evaluated in this code? [1 point]
2. Fine-tune the models, using only 1/10 of the training dataset, with your choice of hyper-parameters. Note: You need to enable the colab's GPU runtime for fine-tuning, and keep the page open for around 20 minutes, so the colab runtime does not disconnect. Try a few different hyperparameters and report the best set, but don't spend too much time here. Report the following:
   (a) Total runtime of a single training run.
   (b) Performance.
   (c) Your choice of hyper-parameters.
   (d) Find 3 examples where the QA model gives an incorrect answer. Suggest one or more reasons why the model failed on the examples.

   Include the notebook (or python code) as you modified it in your submission. [5 points]

4 Exploration of Text Generation Parameters and Probabilities (12 points)

Review the Huggingface documentation on performing auto-regressive text generation, which you encountered a little in Assignment 3, which can be found here. In particular take a copy of the code titled Multinomial Sampling.

Make sure you can run that code, which downloads a (medium-sized) GPT2 model and uses it to generate text given the specific input prompt.

The `model.generate` function (which has the same goal as the generate function you worked with in Assignment 3) has many parameters, including temperature and top_p as described in class. These parameters influence the sampling of the output probabilities which are used to select each word in turn in the auto-regressive generation. There are two other parameters to become familiar with which control how many tokens are generated: `max_length` and `stopping_criteria`. There are also parameters like `repetition_penalty` which penalize repeated words, and reduce their occurrence if the penalty is set above 1. You can review all of the parameters of the `generate` function here.

Consider the following input sequence: "It is important for all countries to try harder to reduce carbon emissions because". In the following steps you are asked to both generate subsequent words from that input context using that GPT2 model, and to explore the probabilities that are generated.

1. To get a sense of the influence of some of the generation parameters, explore at least 10 combinations of temperature and top_p to generate a maximum of 30 tokens using auto-regressive generation with the GPT2 model. Comment on how the generation differs across the range of parameters that you have selected. You must choose your own range, and you'll have to do some exploration to do that; you are free to explore other parameters if you wish. [5 points]
2. Modify the code to output the probabilities of the each word that is generated. You'll need to set these two generate parameters:`return_dict_in_generate=True` and, `output_scores=True`, and extract the probabilities that come in the returned dictionary one call at a time. Provide a table that shows these probabilities, similar to Assignment 3. Comment on the probabilities. [2 points]
3. Write new code that generates the probability tree (like the one drawn on the board in Lecture 6) using the `treelib` package that you can find here. Generate the tree for the above sequence as input, providing the top 3 probabilities for each word position, as far as is practical to see. (You'll have to apply some common sense here to visualize the tree). Comment on the what you see in the tree. Is the tree affected by the top_p parameter or the temperature parameter? Why or why not? Submit your full code that runs the generation and builds and outputs the tree in the file `A4_4_3.py` [5 points]

5 Question Answering using zero- and few-shot prompting with a pre-trained language model (7 points)

To answer this part of the assignment, you should sign up for the OpenAI GPT-3 [1] playground at here: https://beta.openai.com/playground, and you will get a certain amount of tokens for free. Use the default generation parameters.

As discussed in class, recent advances in large pre-trained language models have shown impressive capabilities. These models can do question answering in the form shown above using essentially a zero-shot approach, and they also have some amount of embedded knowledge that allows them to answer a question without context. However, it sometimes takes some careful 'prompt engineering' to be successful. Here are some questions to guide your exploration of the very powerful GPT-3 model:

1. Using the second QA problem from Section 1 (ID 56fa3d788f12f319006300ff), find a way (i.e. engineer the input) to get GPT-3 to answer this question (with all the information given) as best it can. Then, pose the same question to GPT-3 *without* giving it the context. Compare and contrast the results to the one you obtained in Section 2. [2 points]

2. As example of using GPT-3 as an all-purpose tool, consider the following problem that arises in automated psychological counseling. We would like take observations/statements of fact that a counselor may perceive about a patient, and turn it into a "more gentle" suggestion that is less direct than simply stating the fact, and would be easier for a patient to take in. For example, the observed fact might be:

   ```
   You don't like being judged by your family.
   ```

   and the softer version might be:

   ```
   You might not like being judged by your family.
   ```

   You are to "design" a prompt for GPT-3 that takes direct statements of fact and turns them into such "softer" statements. Here is one more example of the original statement: `You're having trouble getting focused.` Try experimenting with different prompts, and the generation parameters (Temperature, top-p, etc.) to see how well you can make such a converter, for this example and one of your own choosing. For both of those examples, report the best 4 results, and show how you achieved them. (i.e. by saying what the prompt and parameters were). [5 points]

References

[1] Brown et. al. Language models are few-shot learners. 33:1877–1901, 2020.

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.