DistilBERT for question answering project documentation

Team members' names and Neptun codes:

Molnár Márk - B44W74; Regényi Ákos - OWPAZM; Sáfrán Gergely- FT6QWV

1 About the chosen topic

Our project's goal is to build a system which can answer user questions related to a text as a given context. This has been done before many times with LLM-s (large language models), and nowadays you can use advanced systems like ChatGPT for the same purpose.

There are mainly two different problems which can be solved when someone builds a system like this. There is in-context question answering and open question answering. The in-context one means that the model is given a text as input and it has to answer questions related to the given text, usually using parts of it as answer. The open question answering problem means something more general, it means you ask something of a model and it gives you an answer related to your question, it is more like two people interacting with each other.
Answering open questions is a more complex problem, it usually requires a complex system and can use larger parameter sized language models (like gpt-3.5-turbo).

2 Our project

In our project we use the DistilBERT [1] language model as a baseline model which is a lightweight variant of the BERT [2] model. BERT is a transformer [3] large language model which was introduced in 2018 by Google researchers and engineers and it is still used today. It is an important model in the line of transformers which has many variants and applications but today is mostly succeeded by other much bigger (both in parameter size and in training corpus) LLM-s like the GPT-3 and GPT-4 models, the LLama model, PaLM and others.

The DistilBERT model is smaller in size[1] compared (it has 40% less parameters than BERT), cheaper and faster to use (60% faster then base-BERT), while maintaining 95% of it's performance. It is a good model to tinker with because you do not need many resources to use it or fine-tune it.

3 Training

We use the baseline model from the DistilBertForQuestionAnswering modul which is called ("distilbert-base-uncased") and furher train it on the SQUAD[2] (Stanford Question Answering

[1] https://huggingface.co/transformers/v3.1.0/model_doc/distilbert.html
[2] https://rajpurkar.github.io/SQuAD-explorer/

Dataset). The dataset has more than 7000 elements, each including a „story” (context), related questions and answers.

The training[3] is done in the train.py file using pytorch and can be containerized. The final model was trained for 9.5 hours in 2 epochs on the dataset, in the google colab environment, mainly because of our limited time and resources for training. Still the model can be used to answer simple questions related to a given text, with a total of 25 and 24.2 exact matches on the validation dataset, and 60.7% and 60.4% F1 score measured in the 2 epochs.
We will evaluate our model using Evaluate library. We use 2 metrics for evaluation „exact match” and f1 score.

Exact Match

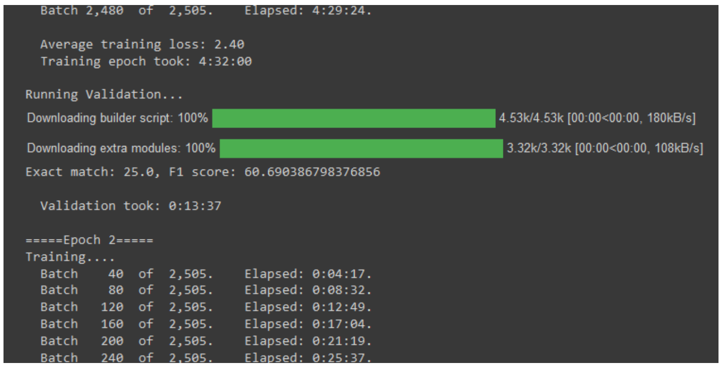
For each question-answer pair if the charecters of the models prediction exactly match with charecters of true answer then EM=1 else 0.When assessing against a negative example, if the model predicts any text at all, it automatically receives a 0 for that example.

F1 score

F1 score depends up on precision and recall and also can be defined with TP, FP, TN, and FN values (true positive, false positive, true negative, false negative).

- $F1\ score = \frac{TP}{TP+\frac{1}{2}*(FP+FN)}$ binary classification
- $makroF1 = \frac{\sum F1\ values}{Number\ of\ classes}$ multi-class classification

If we take the theoritical answers and predicted answers,the number of shared words between theoritical and predicted answer is the basis for f1 score.precision is the ratio of the number of shared words to the total number of words in the prediction, and recall is the ratio of the number of shared words to the total number of words in the ground truth.



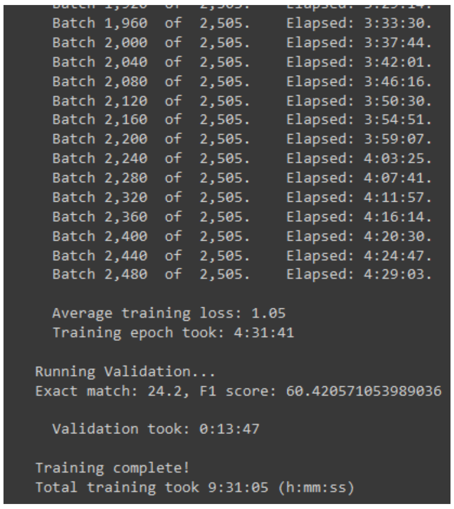
```
  Batch 2,480  of  2,505.    Elapsed: 4:29:24.

  Average training loss: 2.40
  Training epoch took: 4:32:00

Running Validation...
Downloading builder script: 100%  4.53k/4.53k [00:00<00:00, 180kB/s]
Downloading extra modules: 100%  3.32k/3.32k [00:00<00:00, 108kB/s]
Exact match: 25.0, F1 score: 60.690386798376856

  Validation took: 0:13:37

=====Epoch 2=====
Training....
  Batch    40  of  2,505.    Elapsed: 0:04:17.
  Batch    80  of  2,505.    Elapsed: 0:08:32.
  Batch   120  of  2,505.    Elapsed: 0:12:49.
  Batch   160  of  2,505.    Elapsed: 0:17:04.
  Batch   200  of  2,505.    Elapsed: 0:21:19.
  Batch   240  of  2,505.    Elapsed: 0:25:37.
```



```
  Batch 1,960  of  2,505.    Elapsed: 3:33:30.
  Batch 2,000  of  2,505.    Elapsed: 3:37:44.
  Batch 2,040  of  2,505.    Elapsed: 3:42:01.
  Batch 2,080  of  2,505.    Elapsed: 3:46:16.
  Batch 2,120  of  2,505.    Elapsed: 3:50:30.
  Batch 2,160  of  2,505.    Elapsed: 3:54:51.
  Batch 2,200  of  2,505.    Elapsed: 3:59:07.
  Batch 2,240  of  2,505.    Elapsed: 4:03:25.
  Batch 2,280  of  2,505.    Elapsed: 4:07:41.
  Batch 2,320  of  2,505.    Elapsed: 4:11:57.
  Batch 2,360  of  2,505.    Elapsed: 4:16:14.
  Batch 2,400  of  2,505.    Elapsed: 4:20:30.
  Batch 2,440  of  2,505.    Elapsed: 4:24:47.
  Batch 2,480  of  2,505.    Elapsed: 4:29:03.

  Average training loss: 1.05
  Training epoch took: 4:31:41

Running Validation...
Exact match: 24.2, F1 score: 60.420571053989036

  Validation took: 0:13:47

Training complete!
Total training took 9:31:05 (h:mm:ss)
```

1. *Training snapshots (elapsed time, measured scores)*

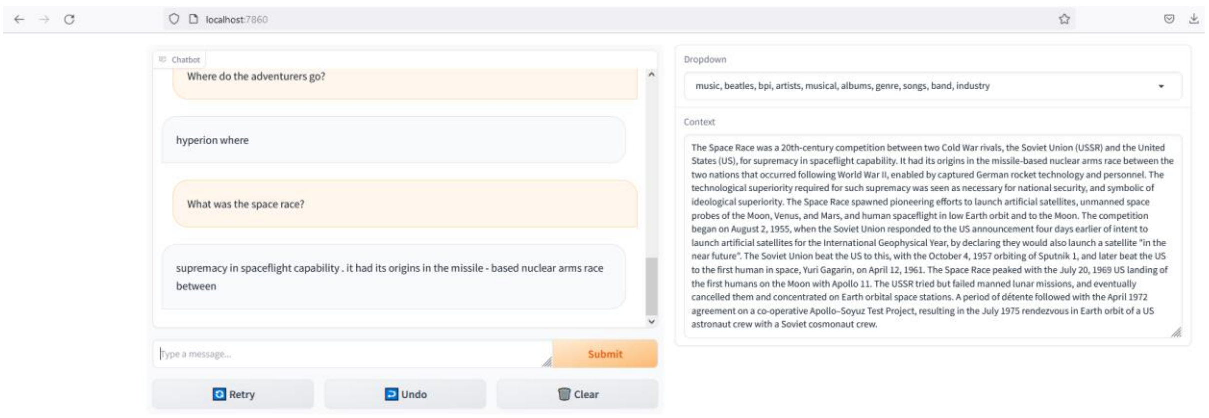
[3] https://www.kaggle.com/code/arunmohan003/question-answering-using-bert

As mentioned above, the model can answer simple questions, mostly those which begin with (where, when, who, why) and the answer is found in the text. It sometimes gives blank tokens as answer mainly because the answer is not in the text or the model was unable to reference it. The trunctuation of the answers is not perfect, sometimes the model goes on with the answer long after the meaningful part.

These problems are less present in a pre-trained model (distilbert-for-question answering trained on squad) which was trained on the same dataset, and we tested it against our model. It gives more precise answers than our model, which better match the questions, and it also gives more correct answers. This shows that with more training there would be better results.

4. Pipeline of the system

After the training was done, we built a pipeline for the application. We use docker containers to build our solution. The containers are using the gradio library to build a web interface which runs on localhost:7860 port.


localhost:7860

Chatbot

Where do the adventurers go?

hyperion where

What was the space race?

supremacy in spaceflight capability . it had its origins in the missile - based nuclear arms race between

Dropdown

music, beatles, bpi, artists, musical, albums, genre, songs, band, industry

Context

The Space Race was a 20th-century competition between two Cold War rivals, the Soviet Union (USSR) and the United States (US), for supremacy in spaceflight capability. It had its origins in the missile-based nuclear arms race between the two nations that occurred following World War II, enabled by captured German rocket technology and personnel. The technological superiority required for such supremacy was seen as necessary for national security, and symbolic of ideological superiority. The Space Race spawned pioneering efforts to launch artificial satellites, unmanned space probes of the Moon, Venus, and Mars, and human spaceflight in low Earth orbit and to the Moon. The competition began on August 2, 1955, when the Soviet Union responded to the US announcement four days earlier of intent to launch artificial satellites for the International Geophysical Year, by declaring they would also launch a satellite "in the near future". The Soviet Union beat the US to this, with the October 4, 1957 orbiting of Sputnik 1, and later beat the US to the first human in space, Yuri Gagarin, on April 12, 1961. The Space Race peaked with the July 20, 1969 US landing of the first humans on the Moon with Apollo 11. The USSR tried but failed manned lunar missions, and eventually cancelled them and concentrated on Earth orbital space stations. A period of détente followed with the April 1972 agreement on a co-operative Apollo–Soyuz Test Project, resulting in the July 1975 rendezvous in Earth orbit of a US astronaut crew with a Soviet cosmonaut crew.

2. Using the gradio web interface

On figure 2 you can see the interface being used. The user can choose a topic from a dropdown list, (each of them represented by 10 words), then a text is chosen randomly in the topic and showed in the „Context" box. There is also a chat interface shown, where the user can submit questions related to the text. By pressing „Submit" the question is shown and the model gives an answer in the same chatbox.

It's important to mention that the texts shown in the interface, and processed by the model are from a different dataset which was never seen by the model. This dataset is called CoQA (A conversational question answering challenge). Because the model was not trained on this dataset, answering questions related to it, is like testing the model in a more real life like scenario, the task is more difficult.

5. Topic modeling

For creating the topics shown above (the 10 words representing one topic), we created a different notebook called „topicmodeling.ipynb”. In this we use the BERTopic model to create the topic labels for the CoQA dataset. We process the „story” records in the dataset which contain all the context examples of the dataset. With some parameter tuning, and reducing the topic number to 10, the final topic labels are created. These are not even close to perfect labels, topic modeling is a difficult task, but they can be used as a baseline solution.

6.How to run it

If you only want to use our trained model you have to use the docker-compose.yml file in the inference container directory, but first of all you need to download the model from our google drive repo. You can find the link in our github repo. If you want to train our baseline model and then you want to use it, then you will need the docker-compose.yml file in the docker directory. To do this, you will need „docker-compose up” command in the correct directory.

7. Conclusions

In this project we trained an instance of the DistilBERT model, to build a pipeline for in-context question answering. The main result of this project is a working pipeline, which is capable of answering questions to a given text, (or in this case a database of texts) which acts similar. The solution can be further improved, by training the model for a longer time and with other datasets, or switching to a pretrained model. The the topic labels can be improved by tinkering more with BERTopic model or using another LLM like ChatGPT. The UI also can be further improved.

Bibliography

[1] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[2] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.