NATURAL LANGUAGE PROCESSING
LAB 4

Master's in Artificial Intelligence & Data Science

Done by:
MOHAMED AMINE FAKHRE-EDDINE,
Supervised by:
Lotfi EL AACHAK

27/05/2024

1 Classification Regression

The goal of this part is to predict the relevance of a a text to a specific topic, using a reccurent neural network. I chose the 'Palestine' topic.

1.1 Scraping

In this first part, it was required to scrape some data from a website, in this case, i scraped the political news from the 'Aljazeera' website, using selenium and beautifulsoup. after scraping a little more than 2000 articles, they were saved in a json file. (2000 is just for demonstration purposes, the scraping can be done for more articles for better results)

1.2 Data Preprocessing

The data was preprocessed by removing stopwords, punctuations, and converting the text to lowercase. The text was then tokenized using the 'nltk' library.

1.3 Scoring articles

The articles were scored using a simple calculation, we calculate the topic and sentence count in the processed text.

$Sentence\_count$

$Topic\_count$

After getting the topic count, we created some statistical features to capture the relevancy of the topic in the given text. Using POS tags, we calculated sentence count, word count, noun count, adjective count, e.t.c. Then we calculated the ratio of topic count to these features. We have used the following statistical features in our model:

- Topic_sentence_ratio: It's a ratio of topic count to the total sentences present in the text.
- Topic_noun_ratio: Ratio of the topic count to the total number of nouns in the text.

These new features helped us find the relevance of that topic in the given news article.

$Noun\_count$

$Topic\_Noun\_ratio = \frac{Topic\_count}{Noun\_count}$

$Topic\_sentence\_ratio = \frac{Topic\_count}{Sentence\_count}$

we produced the Relevance score using a mathematical formula that incorporates all of the statistical data developed in earlier steps. After analysing all of the articles for relevant and irrelevant classes, we set our criterion at 0.4. We normalised the Relevance score such that it always ranged from 0 to 10.

$$Relevance\_score = \begin{cases} 0, & \text{if } R = 0. \\ 10 \times log_{10}(R), & R \neq 0. \end{cases} \tag{1}$$

with $R = Topic\_count \times Topic\_Noun\_ratio \times Topic\_sentence\_ratio \times 10^5$

After calculating the relevance score, we saved the scored articles in a json file.

1.4 Word Embeddings

I used a pre-trained GloVe model to get the word embeddings. The GloVe model had a vocabulary of 1538616 words and a vector size of 256.

1.5 Models

The goal of this part is to predict the relevance of a text to a specific topic, using different variants of RNNs. I implemented the following models:

- Simple RNN
- Bidirectional RNN
- LSTM
- GRU
- LSTM with Attention

Here there is two ways to implement the models, the first one is to get the word embeddings from the GloVe model, generate the word embeddings for our dataset, and then train the model. The second way is to use the GloVe model as a layer in our model, then we should do a specific preprocessing to the GloVe model, like generating the words index, and the embedding matrix. Then we can index the words in each row of our dataset, and feed them to our RNN model.

1.6 Evaluation

A simple training and evaluation were done, but extensive hyperparameter tuning is required to get the best results, which i didn't do due to computational limitations and time constraints.

2 Fine-tuning GPT-2

In this part, we fine-tuned the GPT-2 model on the 'nvidia/HelpSteer' dataset. HelpSteer contains 37, 120 samples, each containing a prompt, a response as well as five human-annotated attributes of the response.

2.1 Data Preparation

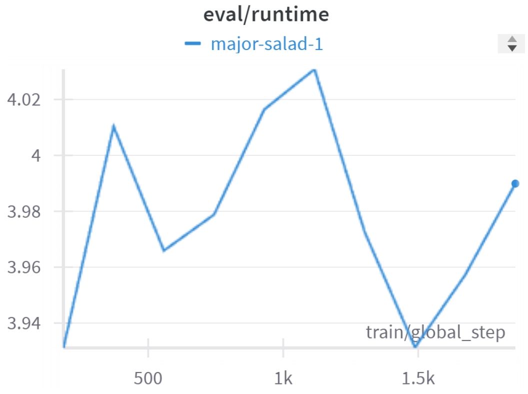
To prepare the data, we generated the token embeddings for texts part of a batch.

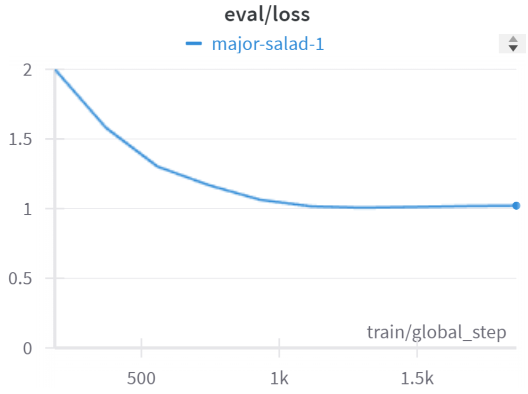
2.2 Model Training

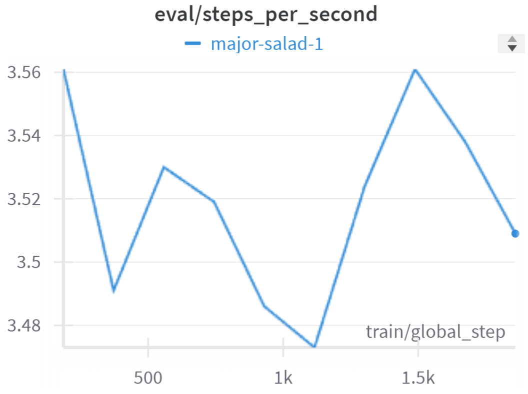
We trained the 'openai-community/gpt2-xl' model leveraging 'PEFT' (Prompt-Encoded Fine-Tuning) on the HelpSteer dataset. We used the 'transformers' library to fine-tune the model.

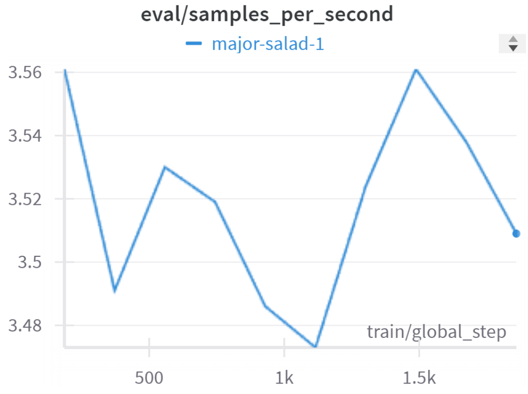
2.3 Evaluation

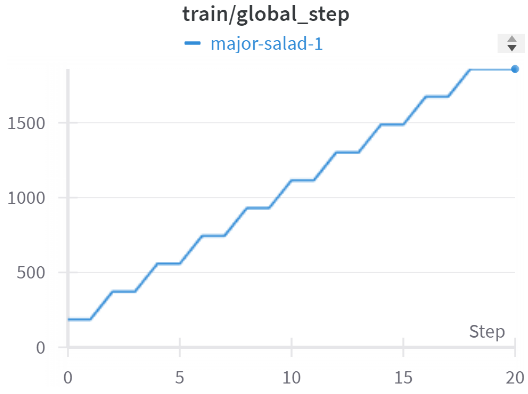
We evaluated the model on the same dataset. Stats

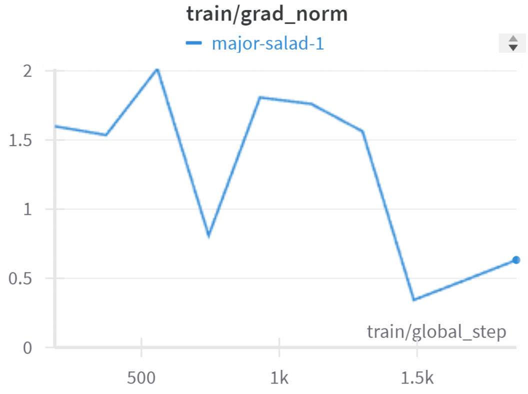


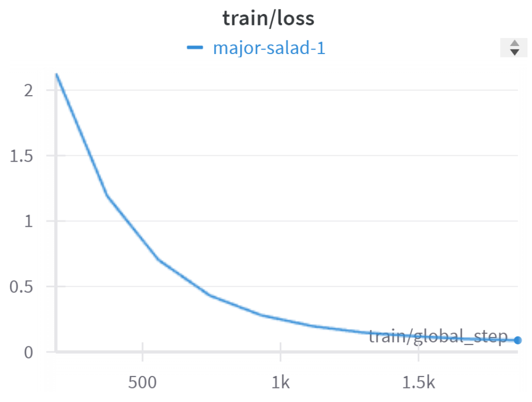


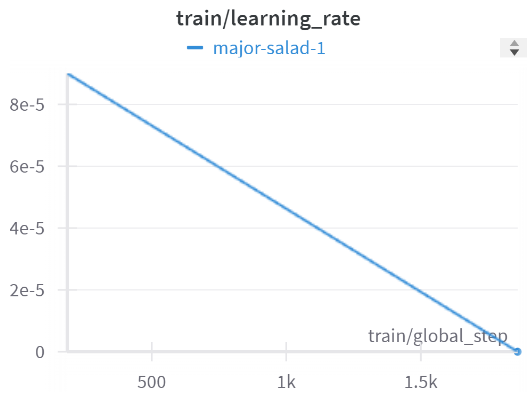


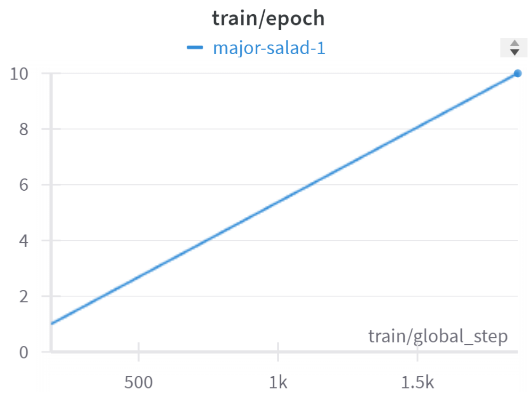


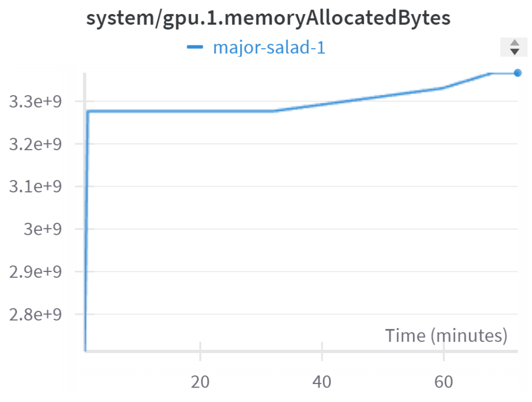


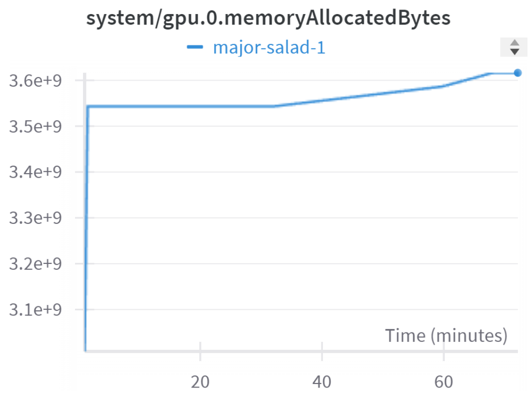


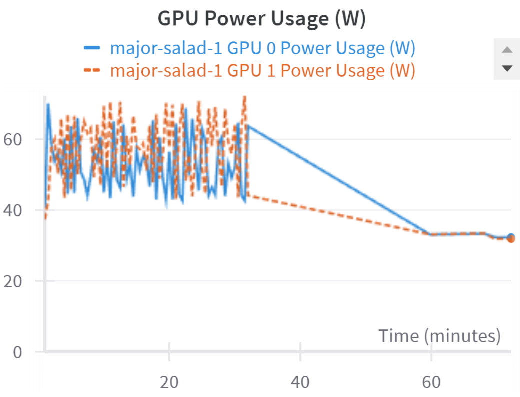

train/loss
major-salad-1
2
1.5
1
0.5
0
train/global_step
500
1k
1.5k

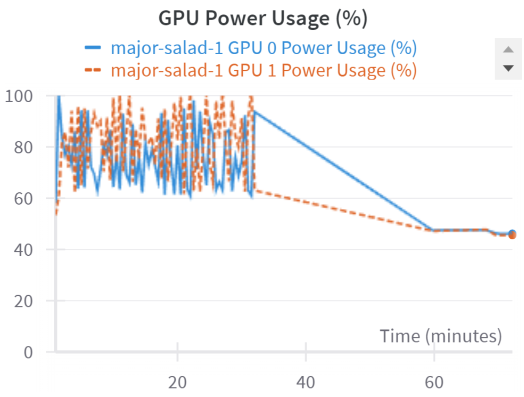

train/learning_rate
major-salad-1
8e-5
6e-5
4e-5
2e-5
0
train/global_step
500
1k
1.5k

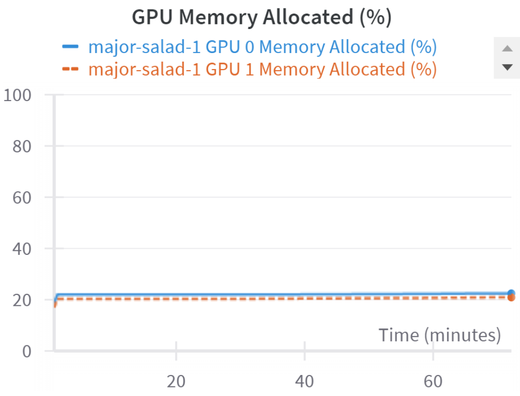

train/epoch
major-salad-1
10
8
6
4
2
0
train/global_step
500
1k
1.5k

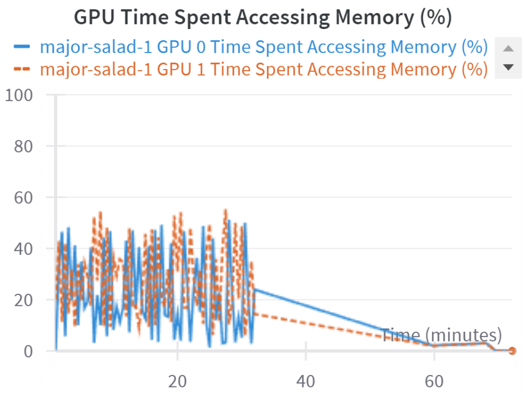

system/gpu.1.memoryAllocatedBytes
major-salad-1
3.3e+9
3.2e+9
3.1e+9
3e+9
2.9e+9
2.8e+9
Time (minutes)
20
40
60


system/gpu.0.memoryAllocatedBytes
major-salad-1
3.6e+9
3.5e+9
3.4e+9
3.3e+9
3.2e+9
3.1e+9
Time (minutes)
20
40
60









3 Fine-tuning BERT on Amazon Reviews

In this part, we fine-tuned the BERT model on a subset of the amazon reviews dataset. I chose the video games reviews.

3.1 Data Preparation

First, we loaded the dataset using specific functions since the dataset is gzipped and not valid jéSon. Then we removed the columns that we don't need, and we kept only the ‘reviewText‘ and ‘overall‘ columns.

I then implemented special classes process and present the dataset in the right format for the BERT model.

3.2 Modeling

I implemented special functions to train the BERT model, in addition to a ‘SentimentClassifier‘ class that inherits from the ‘torch.nn.Module‘ class to classify the sentiment of the reviews.

3.3 Model Training

I fine-tuned the ‘bert-large-uncased‘ model on the video games reviews dataset.

3.4 Evaluation

Evaluation was not done due to computational limitations and time constraints.

References

[1] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.

[2] Stanfordnlp, “Github - stanfordnlp/glove: Software in c and data files for the popular glove model for distributed word representations, a.k.a. word vectors or embeddings.” [Online]. Available: https://github.com/stanfordnlp/GloVe

[3] Z. Wang, Y. Dong, J. Zeng, V. Adams, M. N. Sreedhar, D. Egert, O. Delalleau, J. P. Scowcroft, N. Kant, A. Swope, and O. Kuchaiev, “Helpsteer: Multi-attribute helpfulness dataset for steerlm,” 2023.

[4] Y. Dong, Z. Wang, M. N. Sreedhar, X. Wu, and O. Kuchaiev, “Steerlm: Attribute conditioned sft as an (user-steerable) alternative to rlhf,” 2023.

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[7] A. L. T. G. at Qatar Computing Research Institute (QCRI), “farasa, the state-of-the-art full-stack package to deal with arabic language processing.” 2020. [Online]. Available: https://farasa.qcri.org/

[8] “Weights and biases.” [Online]. Available: https://wandb.ai/site