4NL3 Assignment 5

Guneev Arora - 400477579

April 7, 2025

Contents

1 First Group

1.1 Topic

Group 1: DOTA 2 Message Classification for Auto-moderation Purposes: Codabench Link Hyperlink

Short Description:
The goal of this task/competition was to take messages from many DOTA 2 games and then classify the messages in eight unique categories.
Within the categories, there were two ways of testing the model, which was accuracy and false report rate. False report rate was an important way of testing as it the messages could be toxic/bad or positive. We not only wanted to try and get the best accuracy but also make sure good and bad messages were not classified in the wrong group.

1.2 Approaches

So, for this task, as it was my first group, I wasn't sure what to base my models on, so I decided to go with the same 3 we tested during assignment 4 as I wanted to check how accurate my results were during the assignment compared to the new task and dataset.

1.2.1 Approach 1

Firstly, I remember being disappointed with the results of zero-shot classification, with both long and short prompts.
Therefore, during my previous assignment I had much better luck with a short prompt, however, this time I had better luck with a longer prompt.
I tried using a small prompt of just asking to classify the message, like last time and got accuracy of approximately 10% which wasn't great but what I expected. I then played around with the smaller prompt and tested a few different ones, until eventually I wanted to try something new as results were not great. Thus, I then used a longer prompt and ended with a better accuracy but however, still wasn't the best as I will discuss in results later.

Another thing I switched and tested was the model I used. I tried 4 different models, such as BERT, GPT 2, DeepSeek and Tiny Llama.
I did try using the different kind of prompts on each model until using the longer

one as mentioned before.

So in the end, for zero-shot classification, I used the DeepSeek model with a longer prompt and got my best results with it.

In terms of using the dataset, it was fairly easy to access the text and test with the validation set using files given by group 1.
The model was given labels as names, so I manually converted each of them to the corresponding number to test.

1.2.2 Approach 2

For my second approach, I wanted to test Logistic Regression or Naive Bayes as I was very surpised and impressed during the last assignment.
So obviously, I ended up testing both of them, while also testing using the bag of words models TF-IDF and Count Vectorizer.
Honestly, that was the main testing I had to do, other than that I also tried a few different max iteration values.
Other than that, I used a very similar code to my assignment 4 and also tutorial code to train and test the model.

For this at the end, after the testing, I end up using Logistic Regression with TF-IDF as it performed the best by a small margin within the different mixes.

1.2.3 Approach 3

Now, for my last approach, I wanted to finetune a pretrained model on the dataset.
In terms of the model, I ended up testing different Models such as Bert, RoBERTa, DeepSeek.
Along with that, I also tested using different hyperparameters for each, testing different, learning rates, different weight decays and lastly, different epochs.

After trying a bunch of different combinations, I used the finetuned BERT model with a learning rate of 2e-5, along with 7 epochs as anything more didn't seem to affect it much more.
I also used weight decay of 0.01 to get the best final results from this approach.

1.3 Best Results

Now, in terms of the results with the 3 different approaches, I obviously used the best one from each three to compare.

Now, as I predicted, zero-shot classification let me down again with a final accuracy of 18% which is not great.
Next, the Logistic Regression model, performed a lot better giving me the best accuracy of 41% which was a lot better but wasn't sure if the finetuned model would outperform it or not.

Lastly, the finetuned model did in fact outperform the Logistic Regression model, giving me the accuracy of 47% when testing which was a lot better.

So as we can tell, my third approach of finetuning BERT gave me the best results, Based on my understanding it is due to the fact that the pretrained BERT model understands basic language better than the other approaches due to having patterns other than the text given to us from the dataset which helped a lot more in this case compared to the previous assignment.

Lastly, during the Codabench testing, I submitted both the Logistic Regression and Finetuned BERT model and noticed that the finetuned BERT model had very similar results to the testing dataset accuracy, however, the finetuned BERT model performed the best in terms of the false report rate too probably due to the same reason as mentioned before.

2 Second Group

2.1 Topic

Group 3: Query Classification: Codabench Link Hyperlink

Short Description:
For this task, we had a dataset with a lot of different queries each with specific subject/topic. These were queries of different school/university type questions we had to label.
There were a total of 7 different topics/labels: "Science", "Math/Problem Solving", "Programming", "Writing", "Social Studies", "Business", "Philosophy"

The main form of testing was using accuracy where I should have made a data split in the training data to test locally but took the easier route and just used the Codabench page to test my accuracy.

2.2 Approaches

Now, for this task, I wanted to try something new while also trying the usual to check the difference in results.

2.2.1 Approach 1

To begin with, I wanted to test making a Feed Forward Neural Network model myself to see how my own network would perform.

So as you would think, I started with a very simple model, with a couple layers and an activation function of ReLU.
This ended up not giving me good results, so I then added another layer, and tried a lot of different activation functions like softmax and sigmoid along with ReLU, but ultimately ended up with a model using leaky ReLU.
Other than that, when testing I tried different amount of layers, added normalization, and ended up adding and trying to use different dropout values such as 0.1, 0.05, 0.2, 0.25 and finally using 0.3.
Another thing I tested were the optimizers, where I tried Adam, SGD and finally ended up using AdamW with learning rate 1e-4.
I also used CrossEntropyLoss as the loss function after a little testing.
Finally, I ended up using 500 epochs of training as until around 450-500 epochs, the loss stopped changing much.

I also used the Bert Tokenizer to keep it simple.
Overall, I ended up trying many different combinations of layers, and everything mentioned above.

2.2.2 Approach 2

Now, I wanted to try a different approach to the zero-shot classification as I wanted to check how it would perform if we used a model that was trained on a similar dataset as I wanted to see if it is possible to improve the accuracy.

So to begin with, I tried a couple similar models PriaPillai/distilbert-base-uncased-finetuned-query from HuggingFace.
Like usual, I tried a couple different kind of prompts as the other method of testing other than the model itself.

This was something new, but using the same ideas to try and test different things as I was curious as to how it would perform and ended up using a shorter prompt here for better results.

2.2.3 Approach 3

Now, as I was not happy with the results from the previous two approaches, I decided to try something that has not let me down yet, so ended up using another finetuned model.

The testing for this was pretty much identical to the first groups finetuned model.
Funnily enough, I again ended up using BERT as it performed the best for this at the end.
However, some hyperparameters were different such as the batch size used and also the epochs used.

Honestly, as I assumed, this approach did not let me down at all like usual.
I will further talk about it in the next results section.

2.3 Best Results

Now, in terms of results, as you know, I tried a Feed Forward Neural Network model first which didn't perform horribly but was not good at all either, giving me an accuracy of 35% which was not the best sadly.

Next, I tried to use a zero-shot classification model that was trained on a similar task, which performed better than groups 1 model, but this was probably due to the dataset being much better.
Either way, this gave me an accuracy of 26% which was worse than the neural network model I created.

Finally, I used the finetuned BERT as mentioned, and it gave me an insanely high accuracy of 72% which was miles better than both the other two approaches.

I think that this was due to the fact that the BERT model, already knows how to deal with and understand questions as it is one of the main uses of NLP and AI models, and then getting trained on the exact dataset we had, made it easy to understand keywords and phrases to help us classify the type of topic each query is.

3 Third Group

3.1 Topic

Group 13: Article Headlines Political Classification Official: Codabench Link Hyperlink

Short Description:
For this task, we were given the task to classify if different headlines in articles are left, right or center aligned.

The main form of testing during these was accuracy between the three classes, I chose this task as I thought it would be interesting and also due to the fact that this had fewer labels compared to the other groups.

3.2 Approaches

Now, for this task, I did what I did last time and tried different but also similar approaches to try and test something new with things I know.

3.2.1 Approach 1

This time, I wanted to try something I know is good to begin with, as I would be able to compare it when I try something new later, however, I still wanted to try something new either way. So, I used a fine-tuned a model already trained on a similar task, which was pretty cool, and I tested the same changes on the hyperparameters like usual when I fine-tune a model.

I did of-course try a few models that were trained on the same task.
I ended up using the premsa/political-bias-prediction-allsides-BERT as it gave me the best results compared to the other similarly trained models.

Furthermore, I ended with a learning rate of 3e-5, gradient accumulation steps of 2, weight decay of 0.01 and a batch size of 16 with 10 epochs of training.

3.2.2 Approach 2

So, for the previous group, I tried a Feed Forward Neural Network model, so I wanted to try something a little more complex for better accuracy.
Thus, I decided to try a Recurrent Neural Network model.
As you'd expect, I again started with a simpler model and then added layers and changed the hyperparameters over time after testing such as dropout. Most of it was done very similarly to the Feed Forward Neural Network model I had created for the previous group.

For tokenizing I used a dictionary along with NLTK, to test and try something new. This was a little more complex than I had expected, but I got it working at the end with the help of ChatGPT to be honest.

Other than that, I was pretty satisfied with the results of the model and ended up using AdamW again with weight decay of 0.01 and learning rate of 1e-3 again.

I of course again tried different optimizers, activation functions and hyperparameters in general when testing and was happy with the results.

3.2.3 Approach 3

Now, as zero-shot classification was never great for me, I ended up wanting to try few shot classification.

So, I ended up feeding it 2 examples after testing from a range of 1 to 6 examples.
Like usual, I tried different prompt lengths and different models. I ended up using model "bucketresearch/politicalBiasBERT" as it worked best with a longer prompt due to the few shot classification working better with it.

3.3 Best Results

Finally, in terms of results, I do think this dataset could do better in terms of labeling as there is a big majority on one label.

However, in terms of results, both the recurrent neural network and the finetuned model performed pretty well.
The Recurrent Neural Network gave me an accuracy of 55% which was not bad and the finetuned model gave me an accuracy of 57% which is slightly better.

Sadly, the few-shot classification model did the same as the zero-shot classification model and performed very poorly with an accuracy of 21%.

Thus overall, I think that either the finetuned model or the recurrent neural network model would be the good approaches for this task in terms of what I had tested.

4 Final Comments & AI Tool Usage

I really enjoyed working on this assignment, and testing many different models along with different hyperparameters within the models.
I do think overall, fine-tuned models are usually the best approach but also wish I tested recurrent neural networks with other groups as it performed pretty well for the third group.

Lastly, I think that zero-shot classification is not the best way for most tasks as it usually gives me bad results and accuracy.

I chose my groups by what seemed interesting to me and also fun to work with and play around with. Sadly, datasets were not that accurate or too small as two out of three groups did not have the best final accuracy.

I also want to say I did use previous assignment code, tutorial code and other code from the internet to help me with this assignment.

For AI tools, I used ChatGPT, I did around 52 queries which each are 4.26g = 221.52 grams of CO2 total with all of them being queries on some coding questions but not directly copying code and some other questions on certain topics.