Fall 2021 ACM Coding Challenge

Logan Jackson

September 5, 2021

1 Original Approach

In the Python solution the approach was originally going to be just to take the entire input.txt file and feed it into the model. But unfortuntely that didn't work. So on the second try I decided to split up the contents of input.txt and pass parts of the file to the model one at a time. This is what I decided to do this is really bad because transformers rely on context of past words in the document to be fed back into the transformer in order for it to learn. But since I am too lazy to write my own transformer I decided to take the laziest approach possible which was previouslly described. Now with this terrible approach the question was how to interpret the outputs in a way that was actually valid. Based on the architecture of a Transformer [2] we know that the output of our pretrained model is probabilities. in the case of distilbert this is the probability the model belives that it is correct, aka it's confidence in it's predictions. Now since we have a bunch of outputs from diffrent parts of the file we need to figure out the overall sentiment of the file. The way I did this I am not sure if it's correct but by summing the probabilities we are able to determine the tendency of the distrobution of confidence, negitive predictions are negitive, and positive predictions are positive, if across the file the model is more confident about positive prediction the value of

$$\bigcup P(x)$$

will be positive otherwise it will be negitive. The approach is potentially problematic due to the question of mutal exclusivity can a sentence be both positive and negitive or true neutral? For the sake of this I assume the model will always predict one or the other never both making the options mutally exclusive. Now for the way that I calculated the confidence of the model

$$\bigcap P(x)$$

This I belive is an invalid approach for the following reasons: because the transformer relies on context from the rest of the file each individal batch sentiment could potentially be affected by previous context this means that the pobabilites of each batch are not independent of one another, this is a problem when dealing with an and operation. The product of all sentiments doesn't repersent the overall confidence of the model because each probability would have to be independent. This problem will most likely be addressed in the next model.

2 Numbers

There are two numbers associated with the model output that I previouslly discussed but as I explained earlier one of those numbers I belive to be invalid so there is only one number to talk about. that is:

$$\bigcup P(x) = 0.022$$

where $P(x)$ repersents the output of each batch. The tendency of the transformer is to have 2.2% more confidence in positive results than negitive results meaning that overall it belives the file to be slightly positive.

3 Potential Fixes

I think this approach is actual garbage, but if you can embed the output of the previous batch with the input of the next batch that would probably work and is probably how it's supposed to work. something like this:

```
previous_context = None
for i in range(len(strs)):
   # second arg optional passing None doesn't do anything
   res = clf(strs[i], previous_context)
   previous_context = res[0]['score']
return res

$ res = {'POSITIVE', 0.899999}
```

4 VADER Approach and Numbers

The pretrained model thinks {'neg': 0.065, 'neu': 0.748, 'pos': 0.187, 'compound': 0.9982} this means 6.5% of the text has a negitive valence score 74% has a neutral valence score and 18% has a positive valence score. the model seems to belive that the text is mostly neutral now the issue here lies in "compound score" on the vaderSentiment github page it says:

```
positive sentiment: compound score >= 0.05
neutral sentiment: (0.05 < compound score > -0.05)
negative sentiment: compound score <= -0.05
```

This is not repersentitive of what compound score actually is while looking at the source code [1] you can find how the compound score is actually calculated, based on our output above it implies that the input is very positive. But based on the way that compound score is actually calculated. We see that this is not the case.

$$\frac{sum_s}{\sqrt{sum_s + \alpha}}, sum_s = \Sigma p_i n_i$$

the problem lies in the way that VADER works and how it handles neutral sentiment. Due to the way valence score works VADER will have the tendency to produce true neutral results, a word will have +4, then another word -4 [1] this is also another issue with VADER it's based on individal words predetermined sentiment score meaning that VADER doesn't adapt sentiment scores based on context this is problematic when dealing with larger inputs where sentiment may be highly varied and context reliant. So back to the issue of our compound score. Since the score is 0.99 you would expect the overall sentiment of the paper to be extremely positive but when looking at the break down of the percent of the paper that is positive negitive and neutral, we can see that the paper is mostly labeled neutral and having more positive than negitive so if anything the overall sentiment of the paper should be slightly positive but not extremely positive.

So you may be wondering how do I know that a score of 0.99 is extremely positive? this is due to the normalization function presented above what the normalization does is effectively "squish" the score to be between -1 and 1, where -1 is very negative and 1 being very positive. In theory this works by having each paragraph be labeled either positive or negitive and if you take the sum of all positives and negitives then squish it to a scale based on the

hyper parameter α you should get the overall sentiment of the paper. The problem lies in the way that it handles neutral scores, because it's a sum neutral scores are entirely ignored so even if a paper is mostly neutral say 99% neutral and 1% positive the compound score will come out to be 0.99 an extremely positive score.

So what does this number actually repersent if it doesn't repersent the overall sentiment of the paper due to the fact that it doesn't take into account neutral scores, and how do we fix this? The first question of what it repersents is pretty easy it repersents the normalized sum of positive and negitive sentiments across an input, once again this is invalid because it doesn't take into account neutral seniments by virtue of it being a sum. Second how do we fix this? Well you can actually weight the neutral scores by averaging the valence scores rather than adding them together.

$$score_{avg} = \frac{sentiment}{numInputs}, compoundScore \frac{score_{avg}}{\sqrt{score_{avg} + \alpha}}$$

The reason we normalize this sum is because if a paper is extremely positive or extremely negitive we will get an average score of > 1 or < -1, thus why we have to normalize the score. This is the main reason why the nltk solution contains a fork of the vader source code because I had to modify the model in order to get a number that was repersentitive of the overall sentiment of the input. After all of this the sentiment score is 0.025 or within ± 0.05 of 0.0 giving us a slightly positive but overall nuetral score.

5 Transformer from Scratch Approach

Yeah so the model took to long to train.

6 Native Bayes Approach

References

[1] E.E. Hutto, C.J. & Gilbert. Vader source code.

[2] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.