Report – Machine Learning Project

Shreyansh Rai and Saket Gurjar

Introduction

Although it has been mentioned in the submitted slides there are certain points that must be mentioned over here for ease of reference.

Preprocessing

So In our earlies iteration we took all basic, standard steps in NLP without thinking about the data too much to get a sort of baseline accuracy with a standard approach and a basic model.

We Stemmed the data, removed non alpha numeric symbols, lemmatization and stop word removal was done on the data to ensure that words are converted to the base or root form and so that the frequencies are calculated accordingly while using a naïve bayes model. Any model would be able to use all these different forms of the same root in a similar context. We also removed stop words as mentioned earlier, since we thought the models would be better of with the crux of the sentence and words like ‘a’ ‘an’ ‘the’ and ‘wh’ question words would be removed leaving adjectives nouns and verbs in the mix.

Why the pre-processing did not work was obvious, since the task was to predict troll questions, removing stopwords would be harmful since most questions that are actually legitimate would have a proper structure, with correct use of “WH” question words and articles. Given this fact it became important to retain the stopwords (as is the case in a lot of NLP oriented tasks). So in the end then we did not lemmatize or stem the data and ended up doing extremely light pre processing :

```
lightpre = re.compile(f'([{string.punctuation}“”¨«»®´·º½¾¿¡§£₤’])')
def LightPreProcessing(s):
  return lightpre.sub(r' \1 ', s).split()
```

and this was it. We did not lemmatise our data since it was time consuming at runtime and did not do anything positive for the accuracy. It is common practice as a rule of thumb to use stemmers and lemmatizers if they help with the accuracy. So we decided not to use it in the end to conserve some time on the testing runs.

Models

We tried a variety of models, and sampling techniques to account for class imbalance

Glove Embeddings and Neural network :

This approach failed outright since Dense NNs need context to produce output in NLP tasks. Glove although a good embedding cannot separate context in two different uses of the same word. That means, irrespective of where the word will occur in a sentence, and the context of it's use it will be assigned the same vector. Eg. Reading a book and book a hotel room in both these cases glove and word2vec embeds are the same output vector unlike transformer embeddings like elmo or bert, which provide a different vector based on word position and context. This makes NNs useless without context sensitive embeddings that account for word positioning. Also since this is not a RNN or LSTM we cannot even make sense of the sentence in the form of the temporal data that it is. That is the sequence matters. Due to all the above reasons, this approach failed.

Naïve Bayes :

This was one of our simplest yet one of the more effective models. It gave an accuracy of around 52% on Kaggle after we adjusted the preprocessing as mentioned above (it went from 33 to 52%). Why this was not as good as the fine tuned LR was simply because Naïve bayes is not going to perform good if there is too much difference in the test troll from the train troll. That is, since troll questions are random questions that are assigned to be troll or not based on content, and there is no fixed pattern to trolling, naïve bayes struggles during validation. Due to significant distribution in attributes of troll questions, this model although turned out not to be the best one.

Logistic regression :

This was the model that we ended up using till the end. As mentioned in the slides as well we did threshold tuning and found out optimal train and test sets using stratified k fold here. We used both count and tfidf vectorizers with an n-gram of 1-4 after tuning the model. Tfidf as mentioned in the viva for checkpoint three was important since there were words that were few and far between and we did not want them to lose importance since they were important in predicting the trolls, as mentioned the data set was skewed and most people did not troll but to identify the trolls, and given that the trolls had a varied vocabulary selection, there were words that appeared very less in the corpus. So to ensure they retained importance, we had to use tfidf instead of a regular count vectoriser.

Ensemble :

We tried making an ensemble with a Naive bayes and Logistic regression, two of our best models, and tried to take these 2 models' output and fed it into a Random forest classifier as input and tried using that as out final prediction but that gave similar results as the logistic regression classifier since the LR seemed to be the dominant feature and the Naive Bayes classifier was not able to predict many statements correctly that the LR has misclassified. Thus making that avenue fruitless. Perhaps more tuning would have made it yield a better result but we did not have enough time to pursue that given the last checkpoint was only a week ago.