BBM406 Report

Ege Uçak - 215274670

November 3, 2017

Part I: Identifying the Problem

1. What is the problem?

The problem is to classify tweets to learn if they are positive, negative or neutral.

2. How can the problem be solved?

The problem can be solved using some kind of machine learning approach.

Part II: Solving the Problem

1. What is my approach?

My approach is to use bag of words model with naive bayes method. I implemented the algorithm in python3. First, I loaded training data and trained the algorithm. This training simply is counting occurences of different word from different classes(bag of words).

In detail, I did this training tweet by tweet, first getting rid of punctiation and converting tweet to lowercase (expecting that it will increase accuracy). I call this process as normalization. After normalization I tokenized tweets, and added those tokens to dictionary, which has a value of word instance. While doing these, I also counted how many of each type we have.

After these parts, I recounted tweets with tf-idf (actually it is optional and can be enabled and disabled from in-code arguments). Tf-idf simply is reducing importance of common words for all classes.

After doing all the counting and some other details about naive bayes, I calculated probability of occurence of every word, for every different class. And this is all the training.

Right after training, I started predicting test data, again normalizing tweets first, then summing logarithms of probabilities of words in a tweet for different classes then picking the class with largest value. And for handling tweets that has words which has not been calculated before, I used something which I call weak probabilities which I calculated after training and before predicting. Those weak probabilities are calculated considering that there is a word with only 1 occurence.

Part III: How to Use the Program?

For running the program, ”python3 main.py” should be written. Grams, and tf-idf should be changed inside the code. They are just above ”main()”.

Program take aproximately 0.2 seconds to run, for given training and test set.

Part IV: Analyzing different settings

The calculated accuracies for different settings, as follows;

Accuracy	1 Word	2 Words	3 Words	Tf-Idf
57.00%	✓			
45.20%		✓		
40.50%			✓	
54.50%	✓	✓		
55.70%	✓		✓	
44.80%		✓	✓	
54.70%	✓	✓	✓	
54.40%	✓			✓
45.60%		✓		✓
34.50%			✓	✓
52.90%	✓	✓		✓
53.40%	✓		✓	✓
44.70%		✓	✓	✓
52.40%	✓	✓	✓	✓

The results I got are actually very interesting. I expected to have highest accuracy with unigram, bigram and tf-idf combination but the highest accuracy I got is with unigram. In my opinion he reason behind this is our relatively small training set. This is also the reason for not having a good pattern in our accuracy. For example accuracy for 1 and 3 words is higher than unigram+bigram and bigram.

And expect a couple of situations, tf-idf affected the accuracy in a negative way, which wouldn't be a case if we had a bigger training data.

Another reason behind these results might be the noise in data.

Part V: How to Improve the Results?

As I mentioned before, the reason behind low accuracy is our small training data. For improving the results, only thing that needs to be done is to use a larger training data.

And besides using a larger training data, the training data needs to be cleaned from noise in data.

Part VI: What I Learned?

In this assignment I learnt the basics of natural language processing, bag of words model, and an application of naive bayes.

Part VII: Theory Questions

1. MLE

I have no idea, I didn't sign up for this.

2. Naive Bayes

Applying the Naive Bayes formula which I get from slides;

$$P(w|c) = \frac{count(w,c)+1}{count(c)+|V|}$$

I applied this formula for both politics and sports;

$$P_{politics} = \frac{1}{2} * \frac{(2+1)}{(8+24)} * \frac{(5+1)}{(8+24)} * \frac{(5+1)}{(8+24)} * \frac{(1+1)}{(8+24)} * \frac{(4+1)}{(8+24)} \approx 1.61e-5$$

$P_{politics}$ has a probability of someting around $1.61e-5$