TTDS: CW2

Arnav Sinha (s1951693)

Introduction

This is a comprehensive report discussing the implementations for coursework 3 of TTDS. It comprises of four main sections titled IR Evaluation, Token Analysis, Topic Analysis and Classification.

The implementation code is divided into 3 major tasks. Section IR Evaluation discusses the implementation of Task 1 in detail and provides an analysis for the best performing system for each metric. Sections Text and Topic Analysis discuss the results obtained from Task 2 in detail. Finally, the section Classification discusses the various models used to improve the baseline model accuracy before finally achieving a 5-6% improvement.

There were multiple challenges during the implementation of this coursework but the cited lectures, piazza and the labs were of great help. There was an exponential learning curve as the implementation reached the final parts mostly due to implementing the ideas discussed in the lectures. By implementing IR evaluations, I got a more concrete idea of how to use each metric and perform tests between different systems to see if the systems are significantly different or not.This would help in deciding if the systems are valuable to use for web-search or not. The text analysis section of the coursework taught me how to analyse a document based on token and topic. I also learned how to choose the relevant method depending on the cases which would be helpful for the final CW.

The final implementation of Text Classification was particularly interesting as I got to play around with the results and accuracies. Machine Learning has not been a strong point for me so I was pretty scared before attempting it but it was still very interesting to tweak the preprocessing steps, use various different features or try different models to improve the accuracy on the baseline model. I was not able to preprocess the hashtags as sometimes particularly for tweets, they contain immense information. Words like #Brexit would be relevant after removing punctuations in the preprocessing step but #blacklivesmatter would be of relevance once separated probably using a hashtag module. Also, the tweets can be processed to include the heading of the site wherever relevant for providing more meanings to the tweet. Preprocessing step in this section does not include stemming as the tweets are already a very small document to work with.

One problem I faced was using the csv.reader module which for some reason did not read all my documents and had to change to the basic readLine() method after hours of debugging.

All the parts of the coursework was implemented as instructed and the output ir_eval.csv file and classification.csv file was checked through the python script provided for the right format of the output.

IR Evaluation

The results of 10 queries in 6 systems from system_results.csv are compared with the true relevant results in qrels.csv based on 6 different metrics namely, Precision for top k retrieved documents for each query(P@10), Recall(R@50), R-precision, Average Precision, Normalized Discounted Cumulative Gain(nDCG@10) and nDCG@20. For each system-query pair, the ground truth relevance(rel_i) and the respective rank of of the document($rank_i$) is used for calculating these 6 metrics. If the document is not found then the rank is taken to be infinity. The formulas used for calculating these metrics are listed below:

- **P@k =** $\frac{\#\ of\ relevant\ documents\ retrieved\ @\ k}{\#\ of\ retrieved\ documents\ @\ k}$
- **R@k =** $\frac{\#\ of\ retrieved\ documents\ that\ are\ relevant\ @\ k}{\#\ of\ relevant\ documents\ @\ k}$
- **R-Precision =** $Precision@r$ where r is the number of relevant documents
- **Average Precision =** $\frac{1}{r}\sum_{k=1}^{k} P(k)\ *\ rel(k)$

  where for each query r = no. of relevant documents,

  n = no. of documents retrieved,

  P(k) = Precision@k,

  rel(k) = 1 if retrieved doc@k is relevant, 0 otherwise.
- **nDCG@k=**$\frac{DCG@k}{iDCG@k}$ where DCG@k = rel_1 + $\sum_{i=0}^{k}\frac{rel_i}{log_2 i}$ and iDCG@k = ideal discounted cumulative gain @k.

The table below shows the mean scores of all the 6 metrics for all the 6 systems:

	P@10	R@50	r-precision	AP	nDCG@10	nDCG@20
System 1	0.39	0.834	0.401	0.4	0.315	0.466
System 2	0.22	**0.867**	0.252	0.3	0.162	0.213
System 3	**0.41**	0.767	**0.448**	**0.451**	**0.405**	**0.501**
System 4	0.08	0.189	0.049	0.075	0.057	0.065
System 5	**0.41**	0.767	0.358	0.364	0.316	0.415
System 6	**0.41**	0.767	**0.448**	0.445	0.385	0.481

Further, to check if the system with the highest score for each metric is statistically significantly better than the second system or not, 2-tailed t-test is used. P-value is inversely proportional to

t-value and t-value = $\frac{S2-S1}{\sigma(S2-S1)}\sqrt{N}$. If t-value is big, it means S2-S1 deviates more from 0 which means that S2 is significantly better than S1. So, if p-value is less than 0.05 then S2 is better than S1.

Observing from the table above, the table below shows the best system, the second best system and the p-value between them through the 2-tailed t-test result:

	P@10	R@50	r-precision	AP	nDCG@10	nDCG@20
Best System	S3, S5, S6	S2	S3, S6	S3	S3	S3
2nd best sytem	S1	S1	S1	S6	S6	S6
p-value	0.751	0.343	0.59	0.676	0.269	0.249

As the second table suggests, the p-values for any of the metrics between the best and the second best system is not significantly different since they are vay greater than 0.05. This means that there is no evidence to neglect the Null-Hypothesis H_0 and all the systems are almost similar which is not surprising as other than System 4, all the systems have high comparable values across all 6 metrics.

Token Analysis

This section discusses the word-level differences in the given three classes OT, NT and Quran and further tries to analyse the top scores to provide some observations (in order to characterize the set of documents in the particular class). The words occurring less than 10 times in each class were disregarded.

The table below shows the top 10 highest occurring words for MI and χ^2 for each of the three classes respectively:

Class	MI	χ^2
OT	israel, lord, believ, god, judah, land, son, torment, faith, hous	lord, israel, god, believ, land, faith, son, torment, hous, thing
NT	jesu, christ, lord, israel, discipl, thing, peopl, land, peter, paul	jesu, christ, lord, discipl, thing, peter, paul, israel, peopl, spirit
Quran	god, muhammad, torment, believ, messeng, israel, quran, revel, unbeliev, guidance	god, muhammad, believ, torment, messeng, revel, quran, unbeliev, guidance, disbeliev

The rankings produced by MI and chi squared were not very different. The rankings of the the words were comparable for both the cases with not more than a difference of a couple of ranks.

Through these rankings, it can be seen that the class “OT” has israel, judah, god, faith and beliefs amongst the other words in the top 10 ranks which gives an idea of what the content of the class is like. Similarly, classes “NT” and “Quran” contain words like jesus, christ, disciple, peter, paul and muhammad, god, torment, quran, guidance respectively which act like keywords to that particular class.

Topic Analysis

(the topics are for random_state = 0)

Class	Topic	Top 10 tokens	Own Label
OT	0 (0.069)	0.079*"lord" + 0.062*"god" + 0.061*"suffer" + 0.058*"told" + 0.039*"eye" + 0.039*"word" + 0.033*"peopl" + 0.033*"pray" + 0.033*"ask" + 0.028*"glori"	Lord’s glorious suffering
NT	6 (0.080)	0.168*"god" + 0.089*"thing" + 0.047*"lord" + 0.029*"peopl" + 0.028*"promis" + 0.027*"work" + 0.025*"earth" + 0.023*"understand" + 0.021*"abraham" + 0.020*"righteou"	Abraham’s righteous promise
Quran	12 (0.091)	0.123*"god" + 0.090*"believ" + 0.061*"truth" + 0.039*"lord" + 0.038*"sin" + 0.036*"heart" + 0.032*"soul" + 0.026*"answer" + 0.025*"save" + 0.019*"peopl"	Soulful sinning

By running an LDA Model on these three classes with 20 topics, primarily the most important words in the particular class can be seen from the word probabilities as seen in the table. One example for similar topics in all the three corpuses was found at random_state = 45 and that is topic 11. Some common words forns with their word probabilities are 0.158*"god" , 0.079*"lord", 0.066*"believ", 0.043*"word", 0.031*"peopl" + 0.028*"fear" + 0.026*"heart", 0.026*"hear", 0.025*"prophet", 0.024*"soul".

While implementing token analysis using MI and χ^2, all the words in the corpus is considered which is not very efficient since the words that occur in all the documents regularly do not provide any advantage in analysis rather use more memory. A word in token analysis might get a very high ranking due to a high MI or χ^2 score but might not contribute to the analysis as it might occur equally likely in all the classes.

Classification

To train the baseline, LinerSVC was used as suggested. The train.tsv file provided was first shuffled randomly and then split into a 80% training and 20% development set. The preprocessing for the baseline only involved link removal and tokenisation. The BOW features were used to train the LinearSVC model of the SVM classifier on the training set with C=1000. The model is not quite satisfactory with approximate

- Accuracy = 0.565 and Macro-f1 score = 0.545 on the developement set

- Accuracy = 0.568 and Macro-f1 score = 0.548 on the test set provided.

The three instances on the development set that the baseline model labels incorrectly are:

True	Predicted	Tweet
positive	**neutral**	Watching David Price and the Blue Jays on Saturday night!! Definitely Jays Fever #letsgobluejays
neutral	**positive**	Al Michaels memorialized Frank Gifford at the Hall of Fame game on Sunday http://t.co/aIKSitYrG2 http://t.co/4XqDqbCTno
negative	**neutral**	watching the Mariners & White Sox...these White Sox uniforms may be the ugliest ever

As observed from the table above:

- Example 1 has keywords like “Fever”, excitement indicators like “!!” a hashtag element “#letsgobluejays” which make it a positive tweet but the classifier probably does not identify the context of the word “Fever” or “!!” used in the sentence since the BOW features are used and the hashtags are not preprocessed for the classifier to train on the “lets go” phrase from the tweet. Since, no negative or positive words are used in the tweet, it is predicted as neutral.
- Example 2 is a rather intuitive case where the prediction of the tweet being positive makes more sense than the true category. Since, the Al Micheals pays homage to a legend in his speech before calling for the NFL game, it is a positive emotion. The prediction might be due to the word “memoralized”. But opening the links and including their headings in the tweets while preprocessing would have made the result more concrete.
- Example 3 again is interesting since the token “ugliest” is obviously a negative word even by the most basic standards and even then the tweet is predicted as neutral.

For improving the model, various methods were tested before deciding the final model. For the final improvement model, the punctuations were removed from the tweets as the tweets contained a lot of punctuations which made each token unique without making much of a difference in the context they are used in. Furthermore, no characters other than the alphanumeric characters were used and case-folding was performed. The dataset was split into 90% training-10% development set and the SVC model from SVM classifier was trained on the training split with C = 500. This time, the normalized BOW features were used and the model showed an approximate improvement of 5% with:

- Accuracy = 0.617 and Macro-f1 score = 0.585 on the developement set
- Accuracy = 0.618 and Macro-f1 score = 0.683 on the test set provided.

There were some other methods also used for improving the accuracy of the model on the test set but they did not perform as well. The model descriptions and their respective accuracies are listed in the table below:

Trial Model Description	Dev		Test	
	Accuracy	**Macro-f1**	**Accuracy**	**Macro-f1**
SVC with C=50 and only removing links and tokenising with normalized BOW features	0.587	0.571	0.585	0.517
SVC with C=50 and preprocessing as explained for final model with normalized BOW features	0.593	0.537	0.598	0.680
SVC with C=500 and only removing links and tokenising with BOW features	0.598	0.543	0.596	0.646
SVC with C=50 and preprocessing as explained for final model with normalized BOW features	0.612	0.578	0.618	0.579
OnevsRestClassifier on SVC with C=500 and preprocessing as explained for final model with normalized BOW features	0.618	0.590	0,619	0.599

The OnevsRestClassifier method and the model using SVC with C=50 with preprocessing as explained for final model and normalized BOW features were comparable in accuracy results with the final chosen model. The final model was chosen only because it's Macro-f1 score is higher than the rest.