TTDS CW2

B110854

December 6, 2021

Abstract

This coursework consists of three parts: IR Evaluation, Text analysis and Text classification. In IR evaluation, I have implemented a complete IR evaluation system that evaluates performance of different IR systems based on a different number of metrics, and ran statistical hypothesis testing to determine best systems according to these metrics. In Text analysis, I have implemented a system that uses Quran, New Testament and Old Testament as three corpus, and performs preprocessing and then computes the Mutual Information and Chi-squared scores, before running a LDA over all data. In Text classification, I implemented a system that translates the same corpus from last part into bag-of-words representation and ran a Support Vector Machine algorithm to classify verses into one of the three categories: Quran, New Testament and Old Testament; lastly, I have examined the performance of the classifier based on validation results and experimented on ways for further improvement before running the experiment over test data.

1 General outcomes

1.1 Codes

Codes are written in a single file that contains eleven functions: EVAL, calculateMI, calculateChi2, computeAverageScore, computeTopTokens, preProcess, preProcessTest, uniqueTerms, uniqueClasses, convertToBOWMatrix and calculateScores. EVAL was used in the IR Evaluation task, the four intermediate functions were used in Text analysis task to compute different scores and find the top result, and the remaining were used in Text classification task which preprocesses the data, build bag-of-words matrix and calculates the performance of the classifier over different metrics. A number of different libraries were used in different tasks: csv, math, random, string, sklearn, scipy, numpy, nltk, and gensim; the entire program takes around 8 minutes to run.

1.2 Learning outcomes

In this coursework, I have learned how to evaluate a system from various angles in the IR evaluation task and solidifies my understanding of the underlying mathematical formulae for these different evaluation metrics from past lectures. In Text analysis task, I have gained an experience in implementing Mutual Information and Chi-squared scores over Python, as I have only done it with pen and paper before, it is also fascinating to see the results coming out from the LDA and to think what do they represent. Lastly, the Text classification task combined my knowledge from previous machine learning courses and this course together, and allowed me to experiment on classification models learnt in the past over textual data.

1.3 Challenges

Interpreting LDA model was very challenging as I have neither used it before or encountered its concept prior to the lectures, I did not know that LDA could result in the same topic number so I had to spent a large amount of time debugging. Furthermore, data splitting in Text classification task has caused a problem to me as I tried to split the data after the bag-of-words matrix has been constructed, rather than splitting it during pre-processing stage, which caused a lot of run-time errors.

2 IR Evaluation

The IR Evaluation system consists of two parts: read csv files and an EVAL function, the former part simply reads two csv files that contains the results of each system and the actual query result. EVAL function calculates six system evaluation metrics for each system over each query, and results were rounded to three decimal places:

1. Precision at 10: precision of the system at a cutoff of 10.
2. Recall at 50: recall of the system at a cutoff of 50.
3. r-precision: precision at a cutoff of r, where r is the total number of relevant document a query has.
4. AP: average precision, calculated as $1/r \sum_{k=1}^{n} Precision(k) \times rel(k)$, where r is number of relevant documents given query, n is number of documents retrieved, and rel is a binary value based on whether k is relevant.
5. nDCG at 10: A normalised discount cumulative gain at a cutoff of 10, calculated as $DCGat10/iDCGat10$, where $iDCG$ is the ideal DCG value for the perfect ranking.
6. nDCG at 20: Similar to nDCG at 10 but evaluated at a cutoff of 20.

Table 1 shows the average scores of 6 systems over 10 queries each, each metric's best score were written in italics and highlighted in bold.

System	P@10	R@50	r-precision	AP	nDCG@10	nDCG@20
1	0.39	***0.834***	0.401	0.4	0.363	0.485
2	0.22	***0.867***	0.253	0.3	0.2	0.246
3	***0.41***	0.767	***0.448***	***0.451***	***0.42***	***0.511***
4	0.08	0.189	0.049	0.075	0.069	0.076
5	***0.41***	0.767	0.358	0.364	0.332	0.424
6	***0.41***	0.767	***0.448***	***0.445***	***0.4***	***0.491***

Table 1: Average scores of 6 systems over 10 queries

A 2-tailed t-test with p-value of 0.05 were carried out to check whether the best system for each metric is significantly better than the second best system, this was calculated using scipy.stats.ttest_ind directly, results can be seen in Table 2.

	P@10	R@50	r-precision	AP	nDCG@10	nDCG@20
Top systems	3,5,6	2,1	3,6	3,6	3,6	3,6
p-value	1	0.703	1	0.967	0.883	0.868
Significant	No	No	No	No	No	No

Table 2: T-tests of top systems under each metric

Since p-values for these results are all much larger than the significance level of 0.05, therefore we can conclude that non of the top system under any metric is better than the second top system; this was expected from the results in Table 1 as different systems wins over different metric so they are expected to be similar in strength, systems 3 and 6 seems to be performing really well as they win under most evaluation metrics, while system 4 is the worst since the scores for this system is consistently low compared to the five other systems.

3 Token Analysis

I have computed the Mutual Information and Chi-squared scores for all tokens over each of the three corpora: Quran, New Testament (NT) and Old Testament (OT), these corpora were pre-processed to

OT		NT		Quran	
Token	MI	Token	MI	Token	MI
jesu	0.0387	jesu	0.0566	god	0.0313
israel	0.0364	christ	0.03449	muhammad	0.0302
king	0.0314	lord	0.0238	torment	0.0206
lord	0.0307	israel	0.0154	believ	0.0202
ot	0.0227	discipl	0.0153	messeng	0.0160
christ	0.0206	peopl	0.0115	king	0.0159
believ	0.0185	king	0.0115	israel	0.0156
son	0.0164	nt	0.0109	quran	0.0147
god	0.0161	ot	0.0109	revel	0.0145
muhammad	0.0161	land	0.0103	unbeliev	0.0131

Table 3: The top ten scoring tokens for each corpus ranked by MI

OT		NT		Quran	
Token	Chi-squared	Token	Chi-squared	Token	Chi-squared
jesu	1334.8698	jesu	2908.4640	muhammad	1667.1794
lord	1213.3494	christ	1697.6845	god	1515.8517
israel	1177.8435	lord	857.4801	torment	1204.0430
king	1044.3327	discipl	778.8955	believ	1197.8308
christ	709.8083	nt	539.6681	messeng	944.7982
god	691.8871	peter	507.3513	revel	846.7443
believ	682.3722	paul	507.3513	quran	814.9187
ot	631.6515	thing	461.7590	unbeliev	763.4217
son	620.2789	israel	458.4990	guidanc	730.7405
muhammad	553.8751	spirit	406.4945	disbeliev	708.9025

Table 4: The top ten scoring tokens for each corpus ranked by Chi-squared

lower cases, stemming was applied using PorterStemmer before all stop-words (in EnglishST) were removed. Table 3 shows the results of the top ten scoring tokens for each of the corpus ranked by Mutual Information, and Table 4 shows the results of the top ten scoring tokens for each of the corpus ranked by Chi-squared value. From these results, we can see that Mutual Information is positively correlated to Chi-squared values, tokens with high Mutual Information tends to have a high Chi-squared value, vice versa; this is somehow expected as the Mutual information tells whether one feature can give people information about the others (term and corpus in our case), and Chi-squared value determines whether there is an association between two features. Chi-squared value differs to Mutual Information in features that appears uncommonly which cannot be seen from the table, a feature appears once in correctly classified data would gain a relatively high score in Chi-squared value but quite low in Mutual Information, this is due to the probability multiplication of the feature occurrence in the latter case.

Furthermore, we see that tokens “jesu”, “christ” and “lord” were ranked highly by both Mutual Information and Chi-squared values in NT and OT, this suggests there are similarities between these two corpora and both of the corpora seems to be talking about stories related to Jesus Christ, where Quran seems to be talking about the God, punishment and beliefs as the highly ranked words for Mutual Information and Chi-squared value metrics are slightly different; these tokens can help us to distinguish between OT and NT to Quran corpora as well, since these highly ranked tokens appear very often in texts, but it would be troublesome to distinguish OT to NT using those, a better alternative is to use the rankings of abbreviates “ot” and “nt” directly.

4 Topic Analysis

A LDA was performed on the entire set of verses from all corpora together (in an order of OT tokens, NT tokens and Quran tokens), it was implemented directly using the gensim.models.LdaModel package and the number of topics was set to 20, Table 5 shows the results from LDA which lists the top 10 tokens with the highest probability in the topic identified to be the highest scored topic, for each of the three corpora.

OT - topic 0		**NT - topic 13**		**Quran - topic 5**	
Token	Score	Token	Score	Token	Score
son	0.103	god	0.147	god	0.110
king	0.091	worship	0.061	believ	0.075
brother	0.048	life	0.047	heaven	0.046
record	0.042	love	0.043	earth	0.043
glad	0.031	call	0.039	lord	0.043
citi	0.029	lord	0.034	peopl	0.032
pharaoh	0.029	told	0.032	hand	0.028
father	0.026	soul	0.031	righteous	0.026
mother	0.024	angel	0.031	heart	0.026
high	0.023	fear	0.024	power	0.023

Table 5: The top ten scoring tokens for the highest scored topic for each corpus

By observing these tokens, I would label each of the topic as the following:

- OT - topic 0: family
- NT - topic 13: religion and spirit
- Quran - topic 5: God and people

LDA gives us a general estimation of the possible topics among all of these corpora, it might not be too obvious in our case as these corpora are all bible texts and they tend to be in a similar tone. If this was a collection of newspaper article, LDA might provide us a better categorisation, it will be easier for us to label topics such as sports, cars, politics etc. In essence, LDA provides grouping of documents (20 groups in our case) in each corpus, and allows us to have some deeper insights about these corpora's topic and to distinguish the main themes between different corpora.

After 30 runs of LDA with 20 topics, no clear dominance of topics can be seen from the results, so I have changed the parameter num_topics to 5 and ran 20 further experiments (changed back to 20 in the final submission). Results observed were somewhat interesting, although from previous experiments and corpora selection we know that OT and NT are more related to each other than Quran, but for LDA, we observe topic 1 occurs more in OT and NT, topic 2 occurs more in NT and Quran, topic 4 occurs more in OT and Quran, a table of highest probability words for different topics from these experiments are shown in Table 6, we can see that tokens from Topic 1 and Topic 2 are very similar; one thing to notice is that some of the tokens such as "jesu" that was ranked highly by Mututual Information and Chi-squared values did not appear under highly ranked LDA categories, I think the reason is that these highly occurring tokens appears frequently in all corpora therefore it make no contribution to topic analysis, hence an experiment on LDA with removed highly occurring words can be carried out in future studies.

Topics	**Tokens**
Topic 1	god, lord, peopl, believ, earth, thing, day, heaven, worship, faith
Topic 2	god, lord, believ, life, day, peopl, thing, faith, live, evil
Topic 4	people, suffer, god, grant, day, king, children, guid, honor, lord

Table 6: The top ten scoring tokens for the most frequent topics

5 Classification

5.1 Baseline

5.1.1 Pre-processing

Pre-processing for the baseline model was simply to read the data, tokenize them, remove punctuation and convert tokens to lower case, since the purpose of a baseline is for a comparison only, complicated methods such as stop words removal and stemming were not used; text data was then split in to a 90% training set and a 10% development set.

5.1.2 Bag-of-words representation

Unique tokens and categories were assigned with an ID, and then a documentId $\times$ tokenId sized sparse matrix were created and constructed with token counts for each unique tokens, an out-of-vocabulary column was added to the bag-of-word matrix, as there is a possibility that words did not occur during training occurs in testing.

5.1.3 Training

During training, a SVM multi-class classifier was used with C value set to 1000, this was taken from sklearn directly. Results from training were excellent as the average precision, average recall and average f1 score of the baseline classifier on the development set have all reached an value of around 0.9 which suggests that SVM is actually already a strong model for text classification tasks.

Three instances from the develop set that the baseline system labels incorrectly are shown in Table 7, one hypothesis that the classifier classified these topics wrongly was the imbalance of topic data numbers, there are much more bag of words features in the OT corpus than the rest, the classifier tends to classify texts to OT class as it has more knowledge about it, especially when these texts all have a large number of stop words which could dominate the representation matrix.

Actual	Predicted	Texts
Quran	OT	'the', 'egyptians', 'said', 'what', 'do', 'you', 'suggest' , 'should', 'be', 'the', 'punishment', 'for', 'the', 'thief', 'if' , 'it', 'is', 'proved', 'that', 'you', 'are', 'lying'
Quran	NT	'we', 'cast', 'him', 'out', 'of', 'the', 'fish', 'unto', 'dry', 'land', 'and', 'he', 'was', 'sick'
Quran	OT	'by', 'the', 'high', 'ceiling', 'heaven'

Table 7: 3 instances from the development set that the baseline system labels incorrectly

5.2 Improvement model

I have experimented different settings trying to improve the performance of the classification model, I first changed the data split for corpora Quran and NT, from 90% training and 10% development to 95% training and only 5% development, since we cannot get more verses from these corpora elsewhere. However, the validation performance did not improve or worsen at all which means our original hypothesis was wrong or a 5% change in training data is not significant enough. Then I have made following changes with reasons listed below:

- Applying stemming and removing stop words, so it can remove some unnecessary dimensions and give a better embedding, however the development set performance got worsened so this was reverted.
- Regularization parameter of the SVM model C was set to 1000 which is a large gap from its default value 1, so I have experimented C with smaller values 1, 3, and 5, non of them gave a better development set result.

- Modified other parameters of the origin SVM model but did not get a better development set performance either.
- Tried different settings of neural networks, modified the learning rate, number of hidden layers and perceptrons, number of epochs etc, but the highest performance achieved was 89% and since it takes a long time to train, I decided not to go with this approach.
- I have experimented different classifiers that are available on sklearn, including Decision Trees, Entra Tree, SGD, Gaussian Naive Bayes, K Neighbours, Logistic Regression, Random forests, Gaussian Process, and tuned parameters, and achieved results ranging from around 34% to 89%, with the SGD classifier being the highest, so this is the model I chose.
- After modifications of parameters, I found that setting loss=modified_huber, alpha=0.0002, tol=1e-7 and max_iter=5000 gave me a comparable results to our baseline model, which performed better by 0.006 on average precision, worse by 0.011 on average recall and 0.003 on average f1 score.

These two models were then evaluated on the test set, and the results are depicted below in Table 8; we see an improvement on average precision by 0.013 and on average f1 score by 0.006 but a decrease in average recall by 0.001. This improved model is not significantly better than the SVM model but similar strength to it, however, the training time using the SGD model is much less than using the SVM model.

Model	**Average precision**	**Average recall**	**Average f1 score**
SVM	0.908	0.889	0.898
SGD	0.921	0.888	0.904

Table 8: Testing performance of baseline model and the improved model