Keyi Liu
s1703084

I. General Introduction

❖ **Implementation code**

✓ Successfully calculated 6 measurements for 6 systems (part 1)
✓ Successfully calculated average value for each measurement for 10 queries in 6 systems (part 1)
✓ Successfully wrote results to *ir_eval.csv* and pass file format check (part 1)
✓ Successfully calculated p-value for comparison between the best and 2nd best system for each measurement, But the codes did not show in *code.py* (part 1)
✓ Successfully computed Mutual Information and Chi-square for each of the three corpora (part 2)
✓ Successfully ran LDA model and calculated average score for each topic (part 2)
✓ Successfully identified top 3 topic for each corpora and top 10 tokens for each topic (part 2)
✓ Wrote results of part 2 to MI_&_ChiSquare.txt and LDA_results.txt (part 2)
✓ Successfully built baseline model with SVM+Binary feature weighting (part 3)
✓ Successfully found worry prediction cases for training set and development set (part 3)
✓ Successfully computed precision, recall and F1 to evaluate prediction on each split dataset (part 3)
✓ Successfully computed macro-precision, macro-recall and macro-F1 for each split dataset (part 3)
✓ Successfully calculated term frequency and document frequency for each split dataset (part 3)
✓ Successfully built improved model with SVM+TF-IDF feature weighting (part 3)
✓ Successfully wrote results of part 3 to *classification.csv (part 3)*

❖ **What I learned**

➡ Gained more experience about refactoring codes and building many reusable function
➡ Understood deeply the drawbacks of different measurements in real system by evaluating 6 systems by different measurements.
➡ Understood how to use two tailed t-test to determine if one system is statistically better than the other.
➡ Understood deeply the logic behind mutual information, chi-square and LDA model and their difference among the three feature selection methods.
➡ Learned how to build classifiers for corpus and the whole process of how to evaluate predictions of model.
➡ Learned the process of how to improve the classifier though experiments

❖ **Challenges faced**

➡ Failed to effectively reduce the running time
➡ Failed to use different models, instead of SVM, for text classification
➡ Failed to read useful information from mis-classifying cases in the development set, and only got the idea of what the true class is and what the predicting class.

II.IR Evaluation

In this project, we used different evaluation measurements to evaluate 6 different systems' performance when searching on the same 10 queries. These measurements are P@10, R@50, r-precision, AP, nDCG@10 and nDCG@20 and the mean scores of the 6 systems for each are shown below (see Table 1). Based on each measurements used above, we selected the best system and the second best system by comparing scores of 6 systems (see Table 2). From two tables we noticed that system 3 might have the best performance among the 6 systems. Since system 3 has the highest score by measuring P@10, r-precision, AP, nDCG@10 and nDCG@20, as for the score of R@50, system 3 has the 3rd best score.

	P@10	R@50	r-precision	AP	nDCG@10	nDCG@20
System1(s1)	0.390	0.834	0.401	0.400	0.363	0.485
System2(s2)	0.220	0.867	0.253	0.300	0.200	0.246
System3(s3)	0.410	0.767	0.448	0.451	0.420	0.511
System4(s4)	0.080	0.189	0.049	0.075	0.069	0.076
System5(s5)	0.410	0.767	0.358	0.364	0.332	0.424
System6(s6)	0.410	0.767	0.448	0.445	0.400	0.491

Table 1: The average score for each measurement for 6 systems

	P@10	R@50	r-precision	AP	nDCG@10	nDCG@20
The Best system	(s3, s5, s 6):0.410	s2: 0.867	(s3, s6): 0.448	s3: 0.451	s3: 0.420	s3: 0.511
2nd Best sytem	s1: 0.390	s1: 0.834	s1: 0.401	s6: 0.445	s6: 0.400	s6: 0.491

Table 2: The best and 2nd best systems for each measurement

We used the two tailed t-test to conclude whether the best system is significantly better than the second best system. The null hypothesis is that the mean of the distribution of the difference is zero and the alternative hypothesis is that that the mean of the distribution of the difference is different. The test statistic used here from lecture 10 is $t = \frac{\overline{A-B}}{\sigma(A-B)} * \sqrt{N}$. Where N is the number of samples, we used σ to calculate the standard deviation of $A-B$. By importing *stats* from *spicy* and calling the function of *stats.ttest_ind to* return the calculated t-statistic, we have the two-tailed p-value between the best system and 2nd best system for each measurements (See Table 3). Since we assume the significance level is 0.05, we cannot reject the null hypothesis for each comparison test (no p-value > 0.975) which means there is **not enough evidence to conclude the best system for each measurement is significantly better than 2nd best system for all the system we evaluated here**.

	P@10	R@50	r-precision	AP	nDCG@10	nDCG@20
Input(a,b)	(s3, s1), (s5, s1), (s6, s1)	(s2, s1)	(s3, s1), (s6, s1)	(s3, s6)	(s3, s6)	(s3, s6)
The calculated t-statistic	0.1423	0.3880	0.3108	0.0425	0.1507	0.1679
The two-tailed p-value	0.8884	0.7026	0.7595	0.9666	0.8819	0.8686

Table 3: The results of two tailed t-test

Further Discussion: We used several different measurements to determine the performance of different query search engines, some measurements having clearly drawbacks when applied to real systems. For example: R@50, r-precision and nDCG are calculated based on true relevant results for a random query, but the true relevant results may be different for different users. As for P@N, even we narrowed the searching results at top N while calculating, the position of each relevant result was not considered. More knowledge should be learned to evaluate the performance of search engines and improve the searching results in time (for example, the retrieve relevance feedback algorithm and PRF algorithm mentioned by Lecture 11).

III.Token Analysis

In this part, we read file of Quran, New Testament and Old Testament separately to obtain the document collection and unique words collection for each. Then we computed Mutual Information (MI) and Chi-square scores for each unique words collection and treated the other two document collections as non-target class at the same time. The formula we used shows below:

$$MI : I(U;C) = \frac{N_{11}}{N}\log_2\frac{NN_{11}}{N_{1.}N.1} + \frac{N_{01}}{N}\log_2\frac{NN_{01}}{N_{0.}N.1} + \frac{N_{10}}{N}\log_2\frac{NN_{10}}{N_{1.}N.0} + \frac{N_{00}}{N}\log_2\frac{NN_{00}}{N_{0.}N.0}$$

$$Chi\_square : X^2(\mathbb{D},t,c) = \frac{(N_{11}+N_{10}+N_{01}+N_{00})\times(N_{11}N_{00}-N_{10}N_{01})^2}{(N_{11}+N_{01})\times(N_{11}+N_{10})\times(N_{10}+N_{00})\times(N_{01}+N_{00})}$$

Table 4 shows the top 10 highest scoring words by the two methods for each document collection. For the results of each method, we can easily find most words in common which means the calculation logic behind these two methods are not independent. The word with higher Chi-square usually scores with higher MI score and Chi-square scores are in much higher scale than MI score. Moreover, we noticed that New Testament and Old Testament have better match for two groups of words than Quran and New Testament.

The main difference between these two methods is that Chi-square method usually gives higher score to rare word if this rare word appears once in the right class, but MI method multiply the joint probability (e.g. N_{11}/N) to reduce the score of rare words to achieve better balance. Since we used very large dataset to calculate MI and Chi-square, as the number of words selected increase, Chi-square method will have similar performance as MI. Thus we obtained the similar group of words from these two methods.

Quran: MI	Quran: Chi-square	OT: MI	OT: Chi-square	NT: MI	NT: Chi-square
'allah', 0.1532	'allah', 7058	'israel', 0.0361	'lord', 1119	'jesus', 0.0645	'jesus', 3268
'god', 0.0251	'punish', 917	'lord', 0.0311	'israel', 1070	'christ', 0.0367	'christ', 1795
'man', 0.0195	'believ', 856	'thi', 0.0295	'thi', 953	'discipl', 0.0180	'discipl', 909
'king', 0.0192	'unbeliev', 811	'king', 0.0292	'king', 884	'lord', 0.0160	'faith', 669
'punish', 0.0180	'messeng', 769	'thou', 0.0226	'thou', 776	'ye', 0.0130	'paul', 588
'israel', 0.0179	'god', 704	'thee', 0.0188	'thee', 633	'israel', 0.0128	'ye', 586
'believ', 0.0168	'beli', 683	'believ', 0.0173	'believ', 600	'faith', 0.0126	'peter', 560
'unbeliev', 0.0166	'guid', 677	'judah', 0.0169	'land', 500	'paul', 0.0118	'lord', 538
'messeng', 0.0151	'disbeliev', 665	'land', 0.0155	'son', 488	'peter', 0.0114	'thing', 525
'son', 0.0144	'vers', 665	'hous', 0.0145	'hous', 481	'thing', 0.0113	receiv', 490

Table 4: The Top 10 words from two methods for 3 classes

IV. Topic Analysis

In this part we imported *Lda* from *gensim.models* and imported *Dictionary* from *gensim.corpora.dictionary* to build LDA model to find 20 topics (20 groups) through all the verses from all three corpora. Then we selected top 3 topics for each corpus and in each topic we selected to report 10 tokens (words) with highest probability. The result of this part is shown below (see Table 5).

From the result we can find there is no common top topic between the three corpora. The words 'god' and 'lord' frequently occur in most topics and with relatively high probability. LDA model is more complex than the MI and Chi-squared method, and it calculates probability with a unigram model. LDA model firstly sets the certain topic among all the corpora and then determines which words belong to this topic by considering inner connection and coherence between topics and words instead of focusing on the order of occurrence of words. In the end, we determine what kinds of topic are most relevant to each corpora. But when we calculated MI and Chi-square, the input of unique words collection was considered separately for each corpus instead of considering all unique words collection from three corpora.

	Quran Top1 Topic	Quran Top2 Topic	Quran Top3 Topic	OT: Top1 Topic	OT: Top2 Topic	OT: Top3 Topic	NT: Top1 Topic	NT: Top2 Topic	NT: Top3 Topic
Lablel Number	7	10	8	11	0	4	19	5	6
Label Name	**Devine Supremacy**	**Gifts of God**	**Betrayal and Punishment**	**Obedience**	**The Promised Land**	**Kinship and Humanity**	**Resurrec--tion**	**Redemp--tion**	**Tranmiss--ion of Divinity**
1st highest	god	spirit	love	son	israel	hous	jesus	god	word
2nd highest	lord	heaven	chief	men	lord	behold	dead	man	speak
3rd highest	voic	earth	wrath	citi	children	lord	bodi	work	great
4th highest	fear	world	put	god	god	mine	midst	sin	mouth
5th highest	angel	lord	babylon	servant	cri	hand	jew	thing	flesh
6th highest	perfect	god	trust	6	sea	daughter	death	side	written
7th highest	master	high	men	heard	heart	son	die	made	thing
8th highest	john	face	commit	year	land	burn	gospel	lord	prophet
9th highest	command	prais	possess	lord	peopl	mother	thing	cut	beast
10th highest	worship	mountain	bound	land	preach	save	men	lie	eat

Table 5: The Top 3 topics and Top 10 words for each topic for each corpora

V.Classification

In this part, we split all the corpora from *train_and_dev.tsv* to training set and development set and use the corpus from *test.tsv* as testing set. Then we trained the training set by SVM model with c=1000 to predict on training set, development set and testing set. The performance of our prediction on each was evaluated by precision, recall, f1-score and calculate macro-average precision, recall and f1-score across all three split sets. The first model we build here was treated as baseline model. Based on prediction performance of baseline model, several experiments were tried to improve to the baseline model including increasing the size of training set. Instead of using stop words and stemming, we used TF-IDF feature weighting instead of binary feature weighting.

After training with baseline model, we predicted on development set. The 3 wrong prediction examples were selected and shown below (see Table 6). Since the unique words collection does not cover all the words from the training set, these missing words might lead to misclassification. Moreover, in the baseline model, we are using SVM+binary feature weighting, so missing words might have certain influences on prediction by considering each verse's length is usually very short.

	True class	Prediction class	Verse
Example 1	OT	NT	['behold', 'thou', 'art', 'answer', 'thee', 'god', 'greater', 'man']
Example 2	NT	Quran	['women', 'grind', 'mill', 'left']
Example 3	Quran	OT	['turn', 'heart', 'eye', 'refus', 'leav', 'insol', 'wander', 'blind']

Table 6: 3 examples of mis-classification on predicting development set

Baseline and Improved model explanation:

To improve the baseline model, several experiments were tried to see how much we can improve. And in the end SVM+ TF-IDF was selected to be the improved model. The result for each experiments is shown below (see Table 7).

- **Baseline model**: The dataset was shuffled and then 20% of them was used as training set and 80% as development set. Train an SVM model with binary feature weighting. All the dataset was applied preprocessing include tokenisation, lowercasing, stemming and stop words removal.
- **Experiment 1**: The dataset was shuffled and then 90% was used as training set and 10% as development set. Train an SVM model with binary feature weighting. All the dataset was applied preprocessing include tokenisation, lowercasing, stemming and stop words removal.
- **Experiment 2**: The dataset was shuffled and then 90% was used as training set and 10% as development set. Train an SVM model with binary feature weighting. All the dataset was only applied tokenisation and lowercasing.
- **Improved model**: The dataset was shuffled and then 90% was used as training set and 10% as development set. Train an SVM model with TF-IDF feature weighting. All the dataset was applied preprocessing include tokenisation, lowercasing, stemming and stop words removal. Term frequency and document frequency were calculated for unique words collection for training set, development set and testing set. Use parameters of term frequency and document frequency to calculate weight for each feature in feature vector:

$$w_{t,d} = (1 + \log_{10} tf(t,d)) \times (\log_{10} \frac{N}{df(t)})$$

Model	Train: Macro-P	Train: Macro-R	Train: Macro-F1	Dev: Macro-P	Dev: Macro-R	Dev: Macro-F1	Test: Macro-P	Test: Macro-R	Test: Macro-F1
Baseline	0.999	0.999	0.999	0.840	0.877	0.840	**0.858**	**0.889**	**0.852**
Experiment-1	0.999	0.999	0.999	0.892	0.916	0.886	0.883	0.919	0.894
Experiment-2	1	1	1	0.954	0.961	0.857	0.952	0.963	0.860
Improved	0.999	0.999	0.999	0.869	0.901	0.875	**0.873**	**0.908**	**0.881**

Table 7: Evaluation on each model

From evaluation on prediction for each model. We can find if we increase the training data size, the performance will be improved (Experiment 1, Experiment 2 and Improved model got higher scores on each measurement). For experiment 2, we did not apply stemming and stop words removal, the dimension of feature vector will be largely increased so that the reduction of dimension has not been achieved. Although we could increase our score on training set (since sometimes the verses are empty or contain only one word if we apply stop words removal), but the improvement on Macro-F1 was the smallest among the three experiments. The improved model uses TF-IDF feature weighting and the training set was increased to 90% from 20%, therefore the macro-F1 was increased by around 0.029. However, if we did not change the feature weighting method (using binary feature weighting) but only increased the training size, the macro-F1 of Experiment 1 was increased as well and even a little bit better than SVM+TF-IDF (Improved model). I still do not understand why this situation happens. I think if the verse (document) to be predicted is much larger in this project, the performance of SVM+ TF-IDF will be much better than SVM+Binary.