TTDS CW2 report

s1519734

06 December 2020

1 General Remarks

I completed all the sections of the coursework. The first part was definitely the most straightforward, as it required me to implement the IR evaluation metrics presented in the lecture. There were no significant challenges here.

The second part was definitely more interesting, as it allowed me to understand the subtle difference between Mutual Information and Chi-Square and how they tie in to topic modelling. Again, this task was pretty straightforward. The single challenge was that at the beginning my LDA model was performing badly because I was running it for only one iteration. By increasing the number of iterations, its performance improved and I managed to get some useful topics out of the model.

The last part was definitely the most challenging. The baseline already had a very good performance, so improving it was not easy. I am still not completely sure what we were allowed to use here or not; as a result, I tried out and presented multiple improvements so the last section will definitely go over the page limit.

What I learned:

- Gained a better understanding of the evaluation metrics for IR systems (by implementing them).
- Gained a much better understanding of the MI and Chi-Square metrics and the difference between them. Learned how to train an LDA model.
- For the "Improvements" section, I had to remember a lot of concepts from IAML and MLP I had forgotten. It also gave me an opportunity to read more on BERT and the tokenizer which is used for it and fine-tune the pre-trained model with a high-level package and Google Colab and evaluate its performance on the task at hand. It also gave me a better idea on what to try out first next time I have a text classification task on hand.

The code in "code.py" was put together hastily from 4 different Jupyter notebooks (to the best of my knowledge, it should work). All the class and method definitions are at the top and left as-is. The code which was used to generate the requested files and uses those classes/methods is at the bottom, and is commented out. It's also split by assignment exercises, so it should be clear what should be "un-commented" in order to generate the files for a particular exercise. If one wants to run the code where I add extra data to the training and dev sets, one should download the KJV Bible from `https://www.kaggle.com/oswinrh/bible`.

2 IR Evaluation

2.1 P@10

Systems 5, 3 and 6 have achieved the same (maximum) average P@10, 0.41. What is more interesting though, is that for the same query, they all have the same P@10 score. This means that their performance according to this metric is exactly the same (corresponding to a p-value of 1). System 1 has achieved the second highest average P@10 score, 0.39. Systems 4, 2 and 6 are not statistically significantly better than system 1 (p-value of 0.751). This is to be expected, since their scores are quite similar for each query.
Investigating further why systems 4, 2 and 6 achieved the same vector of scores for the queries, we see that for each query they all happened to retrieve the same number of relevant documents in the first 10 retrieved documents. Naturally, this explains why their P@10 scores are identical for each query.

2.2 R@50

System 2 achieved the highest average R@50: 0.867, while system 1 achieved the second highest average score: 0.834. Judged on the R@50 scores obtained for each document, system 2 is not statistically significantly better than system 1 (p-value = $0.34 > 0.05$). This result makes sense, since the only difference in their performance is that for query 7, system 2 retrieved all relevant documents (in the first 50 docs) while system 1 missed one document.

2.3 R-precision

Systems 6 and 3 both achieved the highest average R-precision: 0.448, while system 1 achieved the second highest average score: 0.401. Systems 6 and 3 had the same R-precision scores for every query (therefore, their performance was identical, p-value=1). Systems 6, 5, and 3 also achieved the same P@10 scores for every query. We can see that system 5's average R-precision was 0.049. This means that even though systems 5, 3 and 6 retrieved the same number of relevant documents in their top 10 retrieved documents, system 5 ranked the relevant documents lower than systems 3 and 6. The result makes sense if we take into consideration that most queries have at most 5 relevant documents. For query number 9, which has more than 10 relevant documents we can see that systems 3, 5 and 6 achieved the same R-precision score, 0.9, while for query number 10 which has 5 relevant documents, system 5's R-precision was 0 while the other 2's R-precision was 0.2.

Systems 6 and 3's R-precision scores were not statistically significantly better than system 1's performance (p-value=0.59 ¿ 0.05).

2.4 AP

System 3 achieved the maximum AP: 0.451 while system 6 achieved the second highest AP: 0.445. There isn't a statistically significant difference between those systems' performance (p-value of 0.67 ¿ 0.05). Their AP scores differ marginally for queries 1, 2, and 9 and are the same for the rest of the queries. The list of retrieved documents are almost identical between the two systems, for every query, so this result makes sense.

2.5 nDCG@10

System 3 achieved the maximum mean nDCG@10: 0.42, while system 6 achieved the second highest mean nDCG@10: 0.4. System 3 was not statistically significantly better than system 6 (p-value of 0.2694). From the high mean AP-scores, we already knew that those two systems retrieve a higher number of relevant documents at higher ranks than other systems. This result also proves that these two systems retrieve more relevant documents at higher ranks than the other systems. The difference in nDCG@10 scores between the two systems is not big, since for each query they retrieve the same documents with minimal differences in order (which is the source of this nDCG@10 difference in mean score).

2.6 nDCG@20

System 3 achieved the maximum mean nDCG@20: 0.511, while system 6 achieved the second highest mean nDCG@20: 0.491. System 3 was yet again not statistically significantly better than system 6 (p-value of 0.24610). Same explanation as for nCDG@10. System 3 continues to marginally outperform system 6 even when calculating nDCG at a lower rank.

3 Token analysis

The mutual information (MI) score[1] between a term and a corpus gives us a measure of how much information we can learn about the corpus given the absence/presence of the term.

The chi-square (Chi2) score[2] between a term and a corpus aims to represent the magnitude of the difference between the actual and the expected counts of the term in the corpus. The higher the chi-square score is, the more confident we can be that there is a relationship between the term and the corpus.

The terms with the highest MI and Chi2 scores for each corpus can be seen in Tables 1, 2, 3, 4, 5 and 6.

[1] https://nlp.stanford.edu/IR-book/html/htmledition/mutual-information-1.html

[2] https://nlp.stanford.edu/IR-book/html/htmledition/feature-selectionchi2-feature-selection-1.html

	Term	MI score
1	allah	0.15319
2	thou	0.03932
3	thi	0.03126
4	ye	0.02849
5	thee	0.02821
6	god	0.02498
7	man	0.01955
8	king	0.01929
9	hath	0.01904
10	punish	0.01801

Table 1: Top 10 words in Quran by MI-score

	Term	Chi2 score
1	allah	7058.784
2	punish	917.837
3	thou	889.245
4	believ	856.012
5	unbeliev	811.822
6	messeng	769.741
7	god	701.822
8	thi	699.436
9	beli	683.328
10	guid	677.282

Table 2: Top 10 words in Quran by Chi2-score

"What can you learn about the three corpora from these rankings?"
We will be using the following abbreviations for the three corpora: Quran=Q, Old Testament = OT and New Testament = NT.

Predictably, for corpus Q the term "allah" has the highest MI and Chi2 scores. This is to be expected, since it has 0 occurrences in the other two corpora and more than 4.5k in corpus Q. We can infer that its presence in a document makes it extremely likely that the document belongs to corpus Q. We can also observe that this term has the highest MI and Chi2 scores in the OT corpus and the third highest MI score in corpus NT. This also makes sense based on its imbalanced counts in three corpora: its presence in a document would heavily imply that the document does not belong in either of the NT or OT corpora.

Several words follow the same trend. For instance, the term "thou" has high MI and Chi2 scores for corpus Q because it doesn't occur even once in

it, but occurs frequently in the other two corpora. From the perspective of a classification problem, its presence in a document would strongly imply that the document does not belong in corpus Q. The terms "thi", "ye", "thee", "god", "man", "king" and "hath" have high MI-scores for corpus Q for the same reason.

The term "allah" has a Chi2 score an order of magnitude higher than the second highest Chi2 score, which stresses its importance for corpus Q. We can notice that the Chi2 top-10 ranking for Q does not contain as many "foreign" terms (with low frequencies in Q but high frequencies in OT, NT) as the MI top-10 ranking. The terms "punish", "believ", "unbeliev", "messeng", "beli" and "guid" all have high frequencies in corpus Q and low frequencies in the other two corpora, thus making them characteristic/important for corpus Q.

	Term	MI score
1	allah	0.08711
2	jesu	0.04087
3	israel	0.03613
4	lord	0.03118
5	thi	0.02951
6	king	0.02918
7	thou	0.02269
8	christ	0.0205
9	thee	0.01886
10	believ	0.01733

Table 3: Top 10 words in the Old Testament by MI-score

	Term	Chi2 score
1	allah	2778.575
2	jesu	1296.973
3	lord	1118.681
4	israel	1070.163
5	thi	953.891
6	king	883.662
7	thou	776.969
8	christ	649.054
9	thee	633.997
10	believ	600.444

Table 4: Top 10 words in the Old Testament by Chi2-score

For the OT corpus, we can notice that the term with the highest MI and Chi2 scores is "allah", for the reasons cited above. The term "jesu" has the second highest MI and Chi2 scores, for similar reasons: it occurs 0 times in the

OT corpus, 33 times in Q and 888 times in NT, making it a good predictor for the NT class (makes sense since Jesus was not referenced in the Old Testament). The term "christ" also has high MI and Chi2 scores for the same reason.

Based on the MI and Chi2 scores of the top terms for the OT corpus and their counts in the three corpora, we can deduce that the terms "israel", "lord", "king", "thou" and "thee" are the most characteristic for the OT corpus. Based on my knowledge of the Old Testament though, the terms "thou"

	Term	MI score
1	jesu	0.06458
2	christ	0.03677
3	allah	0.01935
4	discipl	0.01802
5	lord	0.01607
6	ye	0.01301
7	israel	0.01286
8	faith	0.01267
9	paul	0.01185
10	peter	0.01145

Table 5: Top 10 words in the New Testament by MI-score

	Term	Chi2 score
1	jesu	3268.989
2	christ	1795.001
3	discipl	909.8
4	faith	669.145
5	paul	588.945
6	ye	586.429
7	peter	560.751
8	lord	538.634
9	thing	525.05
10	receiv	490.809

Table 6: Top 10 words in the New Testament by Chi2-score

Similar to the two discussions above, we can infer that the terms "jesu", "christ", "discipl", "faith", "paul", "peter" are characteristic terms for the NT corpus. The terms "allah" and "israel" have high MI scores for the NT corpus, because they are characteristic for the Q and OT corpora, respectively. Even though the term "ye" has the same meaning as the stop-word "you", removing it might not be a good idea. We can see that it has a high Chi2 score for the NT corpus, the reason for this being that it has a reasonably high frequency in it, but occurs 0 times in the Q corpus. Therefore, if it were present in a document,

it would make it more likely that the document does not belong to the Q corpus.

What differences do you observe between the rankings produced by the two methods (MI and Chi2)? Both measures are similar in the sense that if a term has a high Chi2/MI score in a corpus, we can infer that it is important for that corpus. What we can't infer solely based on those scores is the direction of this importance: is the term's absence important or its presence?. To understand this, we need additional information such as the counts of the term in the corpora or our knowledge about the corpora. Without having this additional information, one could try to make guesses. For instance, we can see that the word "christ" has a high MI score for the OT corpus, but it has a higher MI score and the second highest Chi2 score for the NT corpus. From this, we could reasonably infer that "christ" is a term that is more characteristic for the NT corpus than for the OT corpus.

For any of the three corpora, we could see that terms which had low frequencies there but high frequencies in any/both of the other corpora and vice-versa tended to have high MI scores. As a result, the MI top-10 ranking for each corpus contained a mix of terms characteristic to the corpus in question and of terms characteristic to the other two corpora. From the Chi2 top-10 rankings we could see that terms of the second type (high frequencies in the given corpus, low frequencies in the other two corpora) tended to have higher Chi2 scores than terms of the first type (high frequencies in the other two corpora, low frequencies in the corpus in question). Therefore, the Chi2 top-10 ranking for a given corpus contained more terms which were characteristic to the corpus then the corresponding MI top-10 ranking.

This is the highest noticeable difference between the two metrics. Both metrics highlight terms whose presence/absence is important for the corpus, but the Chi2 metric tends to give a higher score to terms whose presence is characteristic to the corpus in question, relative to the other two corpora (aka terms which make the corpus unique relative to the other two).

4 Topic analysis

Create a table of tokens for each top topic and corpus.
The tables can be seen below:

	Term	Probability
1	allah	0.088
2	receiv	0.081
3	gate	0.053
4	believ	0.043
5	fear	0.04
6	brother	0.039
7	field	0.038
8	enter	0.037
9	abraham	0.031
10	wall	0.029

Table 7: Top 10 terms in the best topic for the Quran.

	Term	Probability
1	god	0.308
2	lord	0.087
3	land	0.065
4	year	0.039
5	hear	0.038
6	egypt	0.025
7	peopl	0.023
8	turn	0.021
9	rejoic	0.02
10	gold	0.017

Table 8: Top 10 terms in the best topic for the Old Testament.

	Term	Probability
1	jesu	0.143
2	faith	0.056
3	eye	0.048
4	mine	0.048
5	thing	0.04
6	discipl	0.034
7	jew	0.03
8	heart	0.029
9	bread	0.024
10	wise	0.022

Table 9: Top 10 terms in the best topic for the New Testament.

State your own labels for the 3 topics. That is, in 1-3 words, what title would you give to each of the three topics?

Top Quran topic: "Abraham and Allah". The topic is quite clearly about the story of Abraham and his sons.
Top Old Testament topic: "The plundering of Egypt". This topic may refer to what happened before the Exodus from Egypt.
Top New Testament topic: "Jesus' teachings". This topic does not seem to refer to anything specific.

What does the LDA model tell you about the corpus?
The LDA model was trained on all the documents of the three corpora, put together. The model attempts to represent the documents as mixtures of topics and topics as mixtures of words. Once trained, we can use the model to retrieve what topics best represent each document, according to a probability score. By adding these scores for all the documents in a corpus, we can find out which topics best represent the whole corpus. That is, we can find out what (mixture of) words best represents each corpus.

Are there any topics that appear to be common in 2 corpora but not the other? What are they and what are some examples of high probability words from these topics? Yes, there are.
Here are some topics which have high scores for the Quran and the OT, but a lower score for the NT:
Topic A: 0.245*"day" + 0.049*"sea" + 0.046*"side" + 0.039*"light" + 0.036*"night" + 0.032*"laid" + 0.031*"joy" + 0.031*"rest" + 0.030*"month" + 0.030*"strong"
Topic B: 0.110*"david" + 0.076*"host" + 0.056*"lord" + 0.044*"suffer" + 0.043*"lift" + 0.041*"coven" + 0.035*"river" + 0.033*"rise" + 0.030*"solomon" + 0.028*"wast"
The second topic makes sense, since Solomon and David were referenced in both of these texts.

Some topics which have high scores for the OT and the NT, but a lower score for the Quran:
Topic C: 0.169*"hous" + 0.078*"time" + 0.069*"holi" + 0.063*"brethren" + 0.059*"lord" + 0.055*"pass" + 0.045*"face" + 0.038*"destroy" + 0.034*"walk" + 0.033*"tree" Topic D: 0.308*"god" + 0.087*"lord" + 0.065*"land" + 0.039*"year" + 0.038*"hear" + 0.025*"egypt" + 0.023*"peopl" + 0.021*"turn" + 0.020*"rejoic" + 0.017*"gold". Topic D is the topic with the highest average score for the OT.

The LDA model seems to also give us an idea of what topics / words best characterise two of the corpora but not the third.

How is this different from the things you learned when analysing the data using MI and Chi2? MI and Chi2 give us the (single) words whose presence or absence best characterise each corpus. The scores retrieved for a given term and corpus tell us (almost) nothing about the scores between the other terms and the corpus.

However, the topics retrieved from the LDA model give us a more nuanced view. Keeping in mind that topics are mixtures of words, the topics retrieved from the model represent nothing more but words which together best characterise a single corpus, words which together characterise two of the corpora better than the third and words which characterise all the corpora. These topics may or may not also represent ideas which are common to one or more corpora or specific to a single corpus.

Naturally and as to be expected, we can find words which had high MI/Chi2 scores for a corpus in the top topic of that corpus (e.g: "allah" and "believ" for Quran, "lord" for the OT, "jesu" and "discipl" for the NT). But we can also observe that the top topics for each corpus tend to not contain words which were highly specific to any of the other corpora, as opposed to the rankings for the Chi2/MI scores. This makes sense, since the top topic for each corpus represents the mixture of words which best characterises that corpus, not the singular words which best differentiate it from the other corpora.

5 Classification

- Need to mention what metric you chose to optimise and why. (Macro-f1 is pretty general).
- How you split the training set into train-dev. For starters, stratified sampling, since we expect the test set to have the same structure.

5.1 Misclassified examples

I have implemented the baseline as described in lab 7 (tokenization = removing numbers and punctuation). The documents in the train set have been randomly drawn (without replacement) from "train_and_dev.tsv". The documents which remained were used as a development/validation set. The train set contains roughly 90% of the documents in "train_and_dev.tsv".

The baseline achieves pretty high scores already. The values of the p-macro, r-macro and f-macro scores on the dev set were 0.936, 0.921 and 0.928 respectively. The baseline model has a relatively good performance for documents belonging in the "Quran" and "OT" classes, but achieves lower performance when classifying documents from the "NT" class (p-nt=0.903, r-nt=0.84). The confusion matrix for the dev set can be seen in Figure 1.

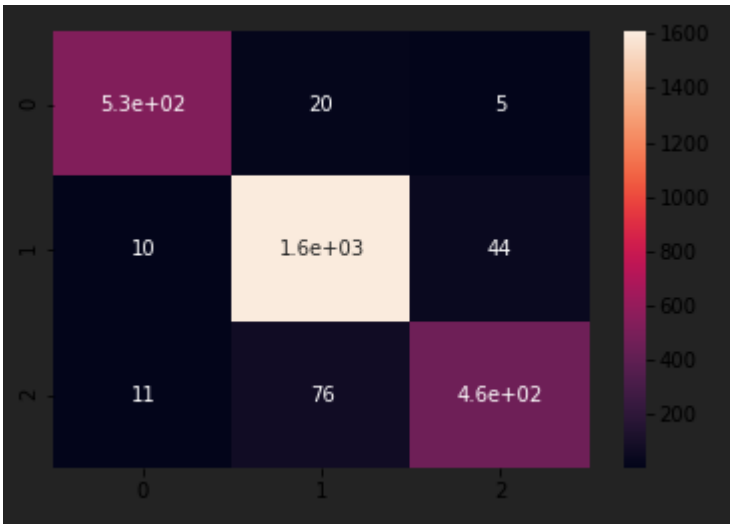

Figure 1: Confusion matrix for baseline predictions on the dev set
Labels on the left represent true labels and labels below predicted labels
0 - Quran, 1 - OT, 2 - NT

From the confusion matrix, we can see that a relatively high number of documents (76) from the NT are misclassified as being from the OT, while 44 OT documents are mistakenly classified as NT. The latter misclassification has a smaller effect on the "p-ot" and "r-ot" metrics since the dataset contains roughly three times more OT documents than NT documents.

Examples of misclassified sentences:

- "Then he called his servant that ministered unto him and said Put now this woman out from me and OOV_term the door after her" - OT doc misclassified as NT.
- "OOV_term was in the days of Herod the king of Judaea a certain priest named Zacharias of the course of Abia and his wife was of the daughters of Aaron and her name was Elisabeth" - NT doc misclassified as OT
- "But last of all he sent unto them his son saying They will reverence my son" - NT doc misclassified as OT.

"OOV_term" indicates that the model encountered a term which was not in the training set. All such terms get the same BOW id (as per lab 7).

The documents in the OT and the NT are much more similar to each other than the documents in the Quran (both in terms of topics and of words used). So, it is to be expected that some sentences such as the third above are misclassified. However, the other two sentences are more interesting: they both contain

terms which were not encountered by the model while training. There is a possibility that those terms are specific to the class of the documents they're a part of.

If we examine the training and dev dataset, we see that the OOV term in the first sentence is "bolt". Indeed, this term occurs a couple of times in the dataset, in two different OT documents, once as a noun and the other as a verb ("bolted"). So, if we were to apply stemming and one of the documents containing this term was kept in the training set, then the other document containing it would have a higher chance to be correctly classified as OT.

The OOV term in the second sentence is "THERE". Even if we were to lowercase all words, this term would likely make little difference to the classification of this sentence, since it's a stopword (occurs frequently in all three classes).

A reason more NT docs are misclassified as OT than vice-versa is the class imbalance of the training set, which contains 16k OT documents, 6k NT docs (and 6k Quran docs). This leads to a bias in the SVM model, towards the class with more documents (OT) which leads to a lower performance on the class with fewer documents (NT) (source: `http://www.cs.ox.ac.uk/people/vasile.palade/papers/Class-Imbalance-SVM.pdf`).

5.2 Improvements

The metric I will try to improve is the "f-macro" measure since a higher value will generally indicate that the model has a good performance for every class. It is also a measure that is suitable to be used when the classes in the training/dev set are imbalanced (so it's perfect for our problem). For reference, the baseline achieved a macro f-score of 0.928 on the dev set and 0.938 on the training set.

5.2.1 Improving partition

The ratio between the number of documents in the three corpora is roughly the same in the given train_dev set as the ratio between the number of verses in the three real texts (Q-5612:OT-16720:NT-5242 in the train_dev set vs Q-6236:OT-23145:NT:7957 = ration of number of verses). This leads us to expect that the same imbalance will be present in the test data as well. Therefore, the number of documents from each class should mirror this ratio in the training set (since the SVC is sensitive to class imbalances). This can be achieved by stratified sampling: instead of sampling 90% of documents from all the documents in the three corpora (which on average will mirror this ratio), we can sample 90% of documents from each corpus (without replacement) and put them together in the train set. The dev set will be formed with the rest of the documents. This simple improvement leads to a score of **0.929** on the dev set and **0.941** on the test set (an improvement of 0.1% and 0.31% respectively, relative to the

baseline).
This partition strategy will be maintained for all other experiments.

5.2.2 Pre-processing experiments

Porter stemming and removing stop-words, just stemming and just removing stop-words were attempted (in addition to tokenizing). For each case, the model had a worse performance on all metrics (dev-set scores of **0.873**, **0.921** and **0.87** respectively). This may indicate that stemming removes meaningful context (aka two words with different endings, each ending being specific to a class may become the same after stemming) and has a destructive influence on the model. Removing stop-words had a minimal (and perhaps statistically insignificant) negative effect on the model's performance. After this experiment, I decided to keep tokenizing as the single pre-processing measure.

5.2.3 More data

The OT and NT verses from the train_dev set are obviously subsets of the King James' version (KJV) corpus. I tried to fix the class imbalance by adding the missing verses to the NT corpus and then I tried to simply add more data by adding the missing verses from both the OT and NT. The scores for the first experiment were **0.929** for the dev-set and **0.948** (1.06% increase) for the test-set while the scores for the second experiment were **0.915** on the dev-set and **0.966** on the test-set (3% increase). Interestingly enough, in these cases the scores for the dev-set are likely a better indicator on the model's performance. Why? Because by adding this data, we are essentially cheating (training on the test set). The test set is formed by the Quran/OT/NT verses which were not in the train_dev dataset (I looked at it after I finished testing all the improvements, naturally) - with the difference that in the test set a single document is sometimes formed from two verses put together (- so it's not exactly cheating). This explains the difference of performance between the test and dev set. Therefore, these results should be discarded. In the following two experiments, the extra documents were removed.

5.2.4 Overfitting

It is quite clear that the baseline model is overfitting, since it has perfect scores across the board for the documents in the training set. We can control this by modifying the regularization parameter "C" (kept to 1000 in all experiments so far). A high "C" value will mean that the SVM margin will be smaller (aka the decision boundaries will likely be highly specific to the training data) and a low "C" value will make the margin larger, which has the chance to improve the generalisation of the model. Therefore, we need to look for smaller "C" values. Indeed, the top scores so far are obtained for C=10: **0.934** on the dev-set (0.65% increase) and **0.943** on the test-set (0.53% increase).

5.2.5 BERT

Finally, using the SimpleTransformers package (`https://github.com/ThilinaRajapakse/simpletransformers`) and a Google Colab GPU, the (smaller) BERT model was fine-tuned to our classification task. We did not experiment with multiple hyperparameter values since training it once took roughly half an hour. The dev-set score was **0.956** (3% improvement) and the test-set score was **0.958** (2.13% improvement). Clearly, BERT generalises much better than the previous SVC models. However, another key difference between this model and our models is the tokenizer used. The BERT model was trained using HuggingFace's "BertTokenizer", which is based on the "WordPiece" algorithm, which (at a very high level) selects the tokens which are the most relevant to the training set. It is unclear whether using this tokenizer is allowed in the context of this coursework (technically, we didn't cover SOTA tokenizers in the course). This gave us another direction for improving our model: feature selection.

5.2.6 Feature selection

We can use the code we wrote for task 2 to compute the Chi-square scores for the terms for each corpus (class). Then, before constructing the bag of words (BOW) model, we can simply remove the words with low Chi-square scores from each corpus. This would hopefully have the effect of reducing the dimensionality of the model and removing words which are useless for our task, thus improving its performance. How do we choose the threshold Chi-squared value? By consulting the table at `https://nlp.stanford.edu/IR-book/html/htmledition/feature-selectionchi2-feature-selection-1.html`, we can choose a significance value and a corresponding threshold. We chose 2.71 as the threshold/critical value. With this preprocessing and C=10 for the SVC model (with other parameters left at their default), we obtain the highest performance yet: **0.976** on the dev-set (5.17% improvement) and **0.981** on the test-set (4.58% improvement). (Note: I am very skeptical of these last results, but I could find no obvious mistake in my code)