COURSEWORK 2

Based on lectures 8,9, 15-18

This coursework is split into three main parts:

1. **IR Evaluation**: based on lectures 8 and 9
2. **Text Analysis**: based on lectures 15/16 and lab 6
3. **Text Classification**: based on lectures 17/18 and lab 7

IMPORTANT DATES

SUBMISSION DEADLINE: SUNDAY, 6TH DECEMBER 2020, 11:59PM

1. IR EVALUATION

In the first part of the coursework, you are required to build a module to evaluate IR systems using different retrieval scores. The inputs to your module are:

1. `system_results.csv`: a file containing the retrieval results of a given IR system and
2. `qrels.csv`: a file that has the list of relevant documents for each of the queries.

Please follow these steps:

- Download the following files: `system_results.csv` and `qrels.csv`. More about the 2 files:
  - `system_results.csv` file, containing the retrieved set of documents for 10 queries numbered from 1 to 10 for each of 6 different IR systems. The format of the files is as follow:

```
system_number,query_number,doc_number,rank_of_doc,score
1,1,710,1,5.34
1,1,213,2,4.23
1,2,103,1,6.21
```

  - `qrels.csv` file, which contains the list of relevant documents for each of the 10 queries. The format of the file is as follows:

```
query_id,doc_id,relevance
1,9090,3
1,6850,2
1,9574,2
```

  where the first number is the query number, the second is the document number and the third is the relevance. e.g. `1,9090,3` means that for query `1`, document `9090` has a relevance value of 3. This value is only important for measures such as DCG and nDCG; while for measures such as P, R, and AP, all listed documents as relevant are treated the same regardless to the value.
- Develop a module `EVAL` that calculates the following measures:
  - **P@10**: precision at cutoff 10 (only top 10 retrieved documents in the list are considered for each query).
  - **R@50**: recall at cutoff 50.
  - **r-precision**
  - **AP**: average precision

  hint: for all previous scores, the value of relevance should be considered as 1. Being 1, 2, or 3 should not make a difference on the score.
  - **nDCG@10**: normalized discount cumulative gain at cutoff 10.
  - **nDCG@20**: normalized discount cumulative gain at cutoff 20.

  Note: Please use the equation in Lecture 9. Any other implementation for nDCG will not be accepted.
- The following file needs to be created in the exact described format.
  - **ir_eval.csv**: A comma-separated-values file with the format:

```
system_number,query_number,P@10,R@50,r-precision,AP,nDCG@10,nDCG@20
1,1,0.00,0.00,0.00,0.00,0.00,0.00
1,2,0.00,0.00,0.00,0.00,0.00,0.00
...
1,10,0.00,0.00,0.00,0.00,0.00,0.00
1,mean,0.00,0.00,0.00,0.00,0.00,0.00
2,1,0.00,0.00,0.00,0.00,0.00,0.00
2,2,0.00,0.00,0.00,0.00,0.00,0.00
...
6,10,0.00,0.00,0.00,0.00,0.00,0.00
6,mean,0.00,0.00,0.00,0.00,0.00,0.00
```

that includes the evaluation results for the 6 systems, labelled with their `system_number` (1-6) which matches the number from `results.csv`.
- Each row should contain a list of the above scores for one of the systems for one of the 10 queries. A full example output file (with, incorrectly, all scores as 0) can also be found here.
- For each system, You should also include a row with the `query_number` set to "mean" for each system that includes the average results for each metric for that system across all 10 queries.
- Before submission, please check that your out files for this file is correct using the Python script.

- Based on the average scores achieved for each system, add a section in your report called "IR Evaluation" to describe the best system according to each score (i.e. what is the best system when evaluated using with P@10, and what is the best system with R@50, and so on). For each best system with a given score, please indicate if this system is statistically significantly better than the second system with that score or not. Please explain why.
  hint: using 2-tailed t-test, with p-value of 0.05. You are free to use existing tool for calculate the p-value. No need to implement this one.
- **NOTE**:
  - All files of results will be marked automatically. Therefore, please be careful with using the correct format.
  - Please round the scores to 3 decimal points (e.g.: 0.046317 --> 0.046).

2. TEXT ANALYSIS

Begin by downloading the training corpora, which contain verses from the Quran and the Bible (split into Old and New Testaments), here. For this coursework, we will consider the Quran, New Testament, and Old Testament each to be a separate corpus, and their verses as individual documents.

Each line in the file contains the `corpus_name` and the `text` which contains the content of a single verse from the corpus, with the two fields separated by a single tab. Complete the following analyses:

- Preprocess the data as usual, including tokenization, stopword removal, etc. Note: you may reuse code from previous labs and courseworks to achieve this.
- Compute the Mutual Information and X^2 scores for all tokens (after preprocessing) for each of the three corpora. Generate a ranked list of the results, in the format `token,score`.
- In your report, add a section called "Token Analysis" to discuss the following:
  - What differences do you observe between the rankings produced by the two methods (MI and X^2)?
  - What can you learn about the three corpora from these rankings?
  - Include a table in your report showing the top 10 highest scoring words for each method for each corpus.
- Run LDA on the entire set of verses from ALL corpora together. Set k=20 topics and inspect the results.
  - For each corpus, compute the average score for each topic by summing the document-topic probability for each document in that corpus and dividing by the total number of documents in the corpus.
  - Then, you should identify the topic that has the highest average score for the three corpora (3 topics). For each of those three topics, find the top 10 tokens with highest probabilty of belonging to that topic.
- Add another section "Topic Analysis" to your report which includes:
  - A table including the top 10 tokens and their probability scores for each of the 3 topics that you identified as being most associated with each corpus.
  - Your own labels for the 3 topics. That is, in 1-3 words, what title would you give to each of the three

topics?
- What does the LDA model tell you about the corpus? Consider the three topics you have presented as well as the other topics and their scores for 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? How is this different from the things you learned when analysing the data using MI and X^2?

3. TEXT CLASSIFICATION

In this part, you are required to apply text classification using the same collection used in the previous section as your training data. The test data will be provided below, 6 days before the deadline. The task is: given a verse, predict which corpus that verse belongs to. Complete the following steps:

- Shuffle the order of the data and split the dataset into a training set and a separate development set. You can split the data however you like. For example, you can use 90% of the documents for training and 10% for testing.
- Apply the steps in the text classification lab to this new dataset in order to get your baseline model: extract BOW features and train an SVM classifier with c=1000 to predict the labels (i.e., which corpus a text belongs to). Note that the input data format is slightly different this time, but you will still need to convert to BOW features. You may reuse your code from the lab.
- Compute the precision, recall, and f1-score for each of the 3 classes, as well as the macro-averaged precision, recall, and f1-score across all three classes. Only train the system on your training split, but evaluate it (compute these metrics) on both the training and development splits that you created (i.e., don't train on documents from the development set).
- Identify 3 instances from the development set that the baseline system labels incorrectly. In your report, start a new section called "Classification" and provide these 3 examples and your hypotheses about why these were classified incorrectly.
- Based on those 3 examples and any others you want to inspect from the development set, try to improve the results of your classifier (you should have already experimented with ways to do this in the lab). You may change the preprocessing, feature selection (e.g., only using the top N features with the highest MI scores), change the SVM parameters, etc. You should create a system that improves over the baseline when evaluted on the development set. However, remember that your final goal is to build a system that will work well on the test set, which is a randomly held-out set of documents from the original copora.
- Now, download the testing set [Available now here]. **Without making any further changes to your baseline or improved models,** train on your training set and evaluate on the new test set you just collected. Report all of your results in a file called `classification.csv` with the following format:

```
system,split,p-quran,r-quran,f-quran,p-ot,r-ot,f-ot,p-nt,r-nt,f-nt,p-macro,r-macro,f-macro
baseline,train,0,0,0,0,0,0,0,0,0,0,0,0,0,0
baseline,dev,0,0,0,0,0,0,0,0,0,0,0,0,0,0
baseline,test,0,0,0,0,0,0,0,0,0,0,0,0,0,0
improved,train,0,0,0,0,0,0,0,0,0,0,0,0,0,0
improved,dev,0,0,0,0,0,0,0,0,0,0,0,0,0,0
improved,test,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

  where the `0` values are replaced with the scores you computed for each class or the macro-average, according to the column names from the header. Each row represents the performance for one of the models (baseline or improved) for one of the splits in the dataset (train, dev, or test).
  in the header, p=precision, r=recall, f=f1-score, quran=Quran, ot=Old Testament, nt=New Testament, macro=macro-averaged scores across all 3 corpora. That is, "p-quran" means the precision score for the Quran class, "r-quran" is the recall for that class, and so on.
  - check the format of your file using this script.
- In your report on this assignment, in the "Classification" section, please explain how you managed to improve the performance compared to the baseline system, and mention how much gain in the Macro-F1 score you could achieve with your improved method when evaluted on the dev set, and how much gain on the test set. Why did you make the changes that you did?
  - Note: it is okay if your test results are different from the development set results, but if the difference is significant, please discuss why you think that is the case in your report.

SUBMISSIONS AND FORMATS

You need to submit the following:

1. **`ir_eval.csv`** containing the IR evaluation scores in the format described above.
2. **`classification.csv`** containing the classification results of the baseline system and the improved system on the train, dev, and test splits as described above.
3. **`code.py`**: a single file including all the code that produces the results that you are submitting and discussing in your report
   - If you will use something other than Python, let us know before submission!
   - Please try to make your code as readable as possible: commented code is highly recommended.
4. **`Report.pdf`**: Your report on the work, no more than 6 pages. It should contain:
   - 1 page on the work you did in the assignment in general, which can include information on your implementation code, summary on what was learnt, challenges faced, comment on any missing part in the assignment.
   - 1 page on the best performing IR system for each score (you can put in a table), and an explanation of if the best system is significantly better than the second system or not, and why.
   - 1-2 pages describing the text analysis including MI, X^2, and LDA results and discussion about them.
   - 1-2 pages on the work you did on classification, and how much improvement you could achieve over the baseline, and how it was achieved (new features? learning method? more training data? ... etc.)

Submit the files listed above on Learn.

Submission deadline: **11:59pm, 6 December 2020**

MARKING

The assignment is worth **20%** of your total course mark and will be scored out of **100 points** as follows:

- **20 points** for the output of the IR Evaluation: `ir_eval.csv`. These marks will be assigned automatically, so you **must** follow the format (remember that you can check that with the provided script).
- **10 points** for the explanation in the report to the best IR system for each score and if it is significant or not.
- **35 points** for your presentation of text analysis results in the report, and discussion about what you learned from them.
- **10 points** for submitting classification results in the correct format with evidence of improvements above the baseline.
- **25 points** for discussion about error cases, what you did to improve on the baseline, and your analysis of the final results.

ALLOWED / NOT ALLOWED

- For the IR measures, scores should be 100% calculated with your own code. It is **NOT** allowed to use ready implementations of these scores. Only for the ttest, you can use libraries (or any tool) to do it.
- For the text analysis, you can use code from the comparing corpora lab. You should **NOT** use existing implementations of mutual information or X^2, but you **are permitted** to use any existing implementation or tool for the LDA-based topic modelling.
- For the classification, you can directly use your work in the text classification lab. However, your mark depends on the amount of work and the improvement you achieve,.

Informatics Forum, 10 Crichton Street, Edinburgh, EH8 9AB, Scotland, UK
Tel: +44 131 651 5661, Fax: +44 131 651 1426, E-mail: school-office@inf.ed.ac.uk