IFN647 – Assignment 2 Requirements

Weighting: 35% of the assessment for IFN647.

Items required to be submitted through IFN647 Blackboard:

1. A PDF or word file includes both
    - Statement of completeness and your name and student ID in a cover page.
    - Solution sections to questions Q1, Q2, Q4 and Q7, and a README section to describe the structure of your data folder setting, the information for "*import* packages" and how to execute your python code in terminal, IDLE or PyCharm as well.
2. Your source code for all other questions, containing all .py files (using a zip file "code.zip" to put them together) necessary to run the solutions and perform the evaluation (source code only, no executables);
3. A demonstration video of how your system works (10 minutes maximum). You can just run your system and record it by using the zoom and save it as "demo.mp4"; and
4. A zip file "result.zip" contains all "result" .dat files (in text).

Please zip all files into a zip file as your "studentID_Asm2.zip" and submit it through the Blackboard before the due date.

Please **do not** include the dataset folder generated by "Unlabelled_datasets.zip" in your submission.

The following are the frameworks/libraries that you can used for assignment 2:

(a) sk-learn
(b) nltk
(c) pandas
(d) numPy
(e) Matplotlib

If you want to use another package or library, you need to get your tutor's approval.

Due date of Blackboard Submission: Friday week 13 (4th June 2021)

Currently, a major challenge is to build communication between users and Web information gathering systems. However, most systems only use queries rather than user information needs due to the difficulty of automatically acquiring user information needs. The first reason for this is that users may not know how to represent their topics of interest. The second reason is that users may not wish to invest a great deal of effort to dig out relevant documents from hundreds of thousands of candidates provided by the system.

In this assignment, you are expected to design a system, "Information Filtering (IF) Model", to provide a solution for this challenging issue. The system is broken up into three parts: Part I (Automatic Training Example Generation), Part II (IF model) and Part III (Testing).

In Part I, the major task is to design an approach to automatically discover a training set for a specified topic (e.g., topic R101, R102, or R150), which includes both Pseudo positive documents (e.g., labelled as "1" if you think they are likely relevant) and Pseudo negative documents (e.g., labelled as "0" if you think they are likely non-relevant). You may need to use the topic title, description or narrative (see the below Example of topic R102), Pseudo-Relevance Feedback technique (or clustering technique) and an IR model for this part to find a training set *D* which includes both D^+ (positive – likely relevant documents) and D^- (negative – likely irrelevant documents) for a given Unlabelled Dataset (*UD*).

Part II is to design information filtering (IF) models to select more terms in *D* and discover weights for them; and then use the selected terms and their weights to rank documents in *UD*.
In Part III, the relevance judgements are provided for all documents used in the unlabelled datasets, and you need to use the relevance judgements to prove your solution is better than an IR model ("the baseline model") which uses only the topic title to rank *UD*.

Data Collection: It is a subset of RCV1 data collection. It is only used for IFN647 students who will be supervised by Prof Yuefeng Li. Please do not release this data collection to other people.

Topic_definitions.txt - It contains definitions for 50 topics (numbered from R101 to R150), where each <top> element (<top>...</top>) defines a topic, including topic number (<num>), title (<title>), description (<desc>) and narrative (<narr>).

Example of topic R102 - "`Convicts, repeat offenders`" is defined as follows:

```
<top>

<num> Number: R102
<title>Convicts, repeat offenders

<desc> Description:
Search for information pertaining to crimes committed by people who
have been previously convicted and later released or paroled from
prison.

<narr> Narrative:
Relevant documents are those which cite actual crimes committed by
"repeat offenders" or ex-convicts.  Documents which only generally
discuss the topic or efforts to prevent its occurrence with no
specific cases cited are irrelevant.

</top>
```

Unlabelled_datasets.zip – It includes 50 Unlabelled Datasets (folders "Training101" to "Training150") for topic R101 to topic R150.

Relevance_judgments.zip – It includes relevance judgements (file "Training101.txt" to file "Training150") for all documents used in the 50 unlabelled datasets, where "1" in the third column of each .txt file indicates that the document (the second column) is relevant to the corresponding topic (the first column); and "0" means the document is non-relevant.

Part I: Automatic Training Example Generation

It requires obtaining a complete pseudo training set ***D*** which consists of a set of positive documents ***D***+; and a set of negative documents ***D***-. In this part, you attempt to present a two approach for finding a complete training set ***D*** in ***UD*** (a given unlabelled document set, e.g., the set of documents in Training102 folder, you can find it in "Unlabelled_datasets.zip"). $\boldsymbol{D} \subseteq \boldsymbol{UD}$, and ***D*** includes at least some likely relevant documents (positive part) and some likely irrelevant documents (the negative part). **The proposed approach depends on the knowledge you have obtained from this unit. You could discuss your approach with your tutor in workshops before you start the implementation.**

Q1) (6 marks) Write a two algorithm in plain English to show your approach for the discovery of a complete training set for 50 topics and the corresponding 50 datasets (Training101 to Training 150). Your approach should be generic that means it is feasible to be used for other topics. For each topic, e.g., Topic R102, you should use the following input and generate the output.

Input: a *topic* (e.g., R102 in "Topic_definitions.txt")
Output: $D = D^+ \cup D^-$, where $D^+ \cap D^- = \varnothing$ and $D \subseteq UD$, where *UD* is a set of documents (e.g., documents in "Training102 "folder, and you can find it in "Unlabelled_datasets.zip").

You need to extract a query *Q* firstly from the topic (e.g., the simple idea is to use <title>, i.e., *Q* = '`Convicts repeat offenders`'; or use all information including the <desc> and/or <narr> elements). Please note you may need to define a query feature function for the query *Q*.

Then use the query *Q* to select sample training examples from *UD*. The following is a possible output *D* (please note it is not the answer) for topic R102 (where D^+ consists of documents with a value of "1" in the third column, and D^- consists of documents with a value of "0" in the third column):

R102 73038 1
R102 26061 1
R102 65414 1
R102 57914 1
R102 58476 1
R102 76635 1
R102 12769 1
R102 12767 1
R102 25096 1

R102 78836 1
R102 82227 1
R102 26611 1
R102 15200 0
R102 13320 0
R102 54745 0
R102 15082 0
R102 53523 0
R102 65306 0
R102 68419 0
R102 29920 0
R102 30456 0
R102 75563 0
R102 28657 0
R102 65394 0
R102 85372 0

Q2) (6 marks) Implement the two algorithm by using Python and test your code using the provided datasets. You also need to discuss the output to justify why the proposed algorithm likely generates high quality training sets. You may use one or two topics and figures to show the justification.

Q3) (3 marks) BM25 based baseline model implementation (see week 9 workshop) – please use the titles as queries (you may set the query feature function $g_i(q_j)=1$ for all query term q_j) to rank documents for each topic, and save the result into 50 files; e.g., *B_Result1.dat*, …, *B_Result50.dat*; where each row includes the document number and the corresponding relevance degree or ranking (in descendent order). The following is the possible result (not the answer) for topic R102:

73038 5.898798484774149
26061 4.273638903483098
65414 4.1414522450167475
57914 3.967136888209526
58476 3.708467957856744
76635 3.5867337114200843
12769 3.4341129093591456
12767 3.352170358051889
25096 2.7646308089876177
78836 2.6823617071618404
82227 2.6056189593652537
26611 2.3595327588643613
24515 2.2258395867976226
33172 2.218657303566887
33203 2.2027873338265396
29908 2.188504022701605
…

Part II: IF model

Q4) (4 marks) Design an information filtering model (i.e., your *IF Model*) that includes both a training algorithm and a testing algorithm *and two information filtering models* in plain English, which illustrates your idea for using your discovered pseudo training set *D* you obtained in Part I to train the IF model. Please make sure that the keywords (terms) selected in *D* should be very useful or important for the given topic.

You will use the following input for the training algorithm to select some useful features as the output:

Input: $\boldsymbol{D} = D^+ \cup D^-$
Output: *Features*

For the testing algorithm, you will have the following input and output:

Input: ***UD*** (e.g., folder "Traning102") and *Features*.
Output: sorted ***UD***

Q5) (6 marks) Implement your two *IF Model* in Python. You need to find useful features (e.g., terms) and their weights for every topic using the proposed training algorithm (in Q4) and store them in a data structure or a file. For all documents in ***UD***, you also need to calculate the relevance score for each document using the proposed testing algorithm; and sort the documents in ***UD*** for each topic according to their relevance scores and save the results into "*IF_Result1.dat*" to "*IF_Result50.dat*" files for 50 topics, where each row includes the document number and the corresponding relevance score or ranking (in descendent order).

The following is the possible result (not the answer) for topic R102:

73038 5.898798484774149
26061 4.273638903483098
65414 4.1414522450167475
57914 3.967136888209526
58476 3.708467957856744
76635 3.5867337114200843
12769 3.4341129093591456
12767 3.352170358051889
25096 2.7646308089876177
78836 2.6823617071618404
…

Part III: Testing

Q6) (5 marks) Implement a python program to calculate top10 precision, recall and F1 (you may use extra measures, e.g., average precision) for both your *IF Model* the Baseline Model by using the provided relevant judgements (see "Relevance_judgments.zip" file in the Blackboard) for all 50 topics and save the results into "*EResult1.dat*" and "*EResult2.dat*" for your *IF Model* and the Baseline Model, respectively.

For example, you should use the following inputs to evaluate your *IF Model* for topic R102:

Inputs: "*IF_Rresult2.dat*" and "*Training102.txt*"

The following is the possible result (not the answer) in the csv file "*EResult1.dat*" (the output of the evaluation for your *IF Model*):

Topic	**precision**	**recall**	**F1**
101	0.130435	0.428571	0.20
102	0.020100	0.029630	0.023952
103	0.046875	0.214286	0.076923
…			

Q7) (5 marks) You will get **5 marks** if you can reject the ***null hypothesis***, "*there is no difference in effectiveness between your IF Model and the baseline model*".

You can choose any measure calculated in Q6, AND use "*t-test*" to answer this question. You need to provide details of the settings for the "t-test" and the explanation of the "t-test" results to prove:

(a) Your *IF Model* is better than the baseline model; and
(b) There is a difference between your *IF Model* and the baseline model.

Please note that if the performance is not satisfactory, you can use the evaluation results to validate your *IF model* by tuning parameter settings or updating your design.

Please Note

- Your programs should be well laid out, easy to read and well commented.
- All items submitted should be clearly labelled with your name and student number.
- Marks will be awarded for programs (correctness, programming style, elegance, commenting) and evaluation results, according to the marking guide.
- You will lose marks for missing or inaccurate statements of completeness, and for missing sections, files, or items.
- Your result does not need to be the same as the sample outputs.

END OF ASSIGNMENT 2