CS-410 Text Information Systems – Fall 2024

HOMEWORK 1

Handed out: September 20, 2024 Due: October 4, 2024 at 23H59 CT

Submission through Gradescope (Problems 1,2,3) and Canvas (Problems 4 and 5): upload the .py files with the names textretrieval.py (Gradescope, i.e., problems 1-3) or HW1-yourNETID.py (Canvas, i.e., problems 4,5). Please, submit your work (2 Python script files) before the deadline (above)

PART I: PYTHON PROGRAMMING (AUTOGRADED - GRADESCOPE)

For this homework, you will use AG's News Topic Classification Dataset. The dataset is available here https://github.com/mhjabreel/CharCnn_Keras/tree/master/data/ag_news_csv (the original dataset is available at http://groups.di.unipi.it/ğulli/AG_corpus_of_news_articles.html). This is a set of more than 1 million news articles. This dataset is also relatively simple to handle and analyze. It has 3 columns, 2 string-type (title, and description) and 1 categorical attribute (class). You will work with the "description" field (third column). For this exercise you can use any library for processing and handling data (Pandas is recommended).

For each model you will implement below (Bit Vector, TF-IDF, and Word2Vec) print the result of the search (the top-5 more relevant documents with their ranking and the bottom-5 documents, i.e., the less relevant documents, and their ranking value).

Since Part I is autograded, you will have access to an skeleton/template code which you will use to create your solution and upload it through Gradescope with the format (including the filename) indicated above.

1 Text Data Parsing and Vocabulary Selection(15 points)

For this exercise, we will use the dataset above. Specifically, use the file
`https://raw.githubusercontent.com/mhjabreel/CharCnn_Keras/master/data/ag_news_csv/test.csv`
Create a function to read the texts of this corpus/collection (where each row corresponds to one document/news source). Remove stop-words, transform words to lower-case, remove punctuation and numbers, remove HTML tags, and excess whitespaces (keeping only one space that separates words). Create a vocabulary (a list of words) using the top 200 most frequent words in the text. Sort them in decreasing order of frequency. Because yuo are using the `test.csv` file, the dataset in this implementation is small and manegeable in a local computer. Feel free to use any library to implement the data access and parsing. See the note at the end of this document about what to do if you require a special library.

The autograder will provide you with immediate feedback for:

```
1.1) Stop words or punctuation (5/5)
1.2) Vocabulary size (5/5)
1.3) Vocabulary query words (5/5)
```

Other tests will run hidden in the background and won't be available to the student since these are run across various sections of the class. You may get deductions depending on those background tests.

2 Document Relevance with Vector Space Basic Model (25 points)

1. Create a script that automates the process of computing word relevances using a VSM using the bit-vector representation. Please see the code skeleton for reference.

2. Test your implementation for the following queries:

- query = "olympic gold athens"
- query = "reuters stocks friday"
- query = "investment market prices"

Immediate feedback will be available for:

```
2.1) text2BitVector representation (10/10)
2.2) bit vector ranking/relevance score computation (5/10)
```

The remaining points will be assigned by a hidden test. Note that a manual grading component will be performed for this problem. If you used a library call instead of implementing the bit vector (representation and scoring) functions, you will get only 10% or 0% (90% or 100% deduction) of the score depending on the level of effort.

3 Document Relevance with Vector Space TF-IDF Model (40 points)

1. Create a script that automates the process of computing word relevances using a vector space representation using the TF-IDF using Okapi-BM25 - without document length normalization.

2. Test your implementation for the following queries:

- query = "olympic gold athens"

- query = "reuters stocks friday"
- query = "investment market prices"

Immediate feedback will be available for:

```
3.1) computation of IDF (10/10)
3.2) text2Tfidf representation (15/15)
```

The remaining points will be assigned by a hidden test. Note that a manual grading component will be performed for this problem. If you used a library call instead of implementing the TF-IDF (representation and scoring) functions, you will get only 10% or 0% (90% or 100% deduction) of the score depending on the level of effort.

PART II: PYTHON PROGRAMMING (CANVAS)

4 Document Relevance with Word2Vec (20 points)

1. Create a script that automates the process of computing word relevances using a vector space representation using word2vec. In this exercise, you don't have to implement word2vec but only use a library. Your job is to use the average log-likelihood of W2V as a relevance function. There are many ways to resolve this as we discussed in class. For instance you can use a BOW representation and use the words in the query (and maybe a vocabulary) to compute the score, or you can use other geometric information of the embeddings. Feel free to use a pretrained W2V model as long as it includes the relevant words. You can reuse the data preprocessing code you used for PART I (Q1)

2. Test your implementation for the following queries:

- query = "olympic gold athens"
- query = "reuters stocks friday"
- query = "investment market prices"

5 (EXTRA CREDIT - OPTIONAL) Document Relevance with Vector Space (15 points)

As in Part I, implement a solution to Problem 3, but this time add document length normalization. Run the tests with the queries in Problem 3 and print the solutions on the screen.

AUTOGRADER NOTE

Platform: The autograder uses Python 3 over Ubuntu 22.04 with the libraries indicated below.

Libraries: If you need a library other than Pandas, NLTK, Numpy, please, send a private note to the instructor and TAs through Campuswire so that they can verify the eligibility of the library to add it to the autograder. Note that you are supposed to implement both the Bit-Vector and TF-IDF representations so you can't use a library that implements them. However, you may need other libraries to access or store data. You don't need a library installed in the autograder for Word2Vec since that corresponds to Part II, which has to be submitted via Canvas.

Coding: Please, use the class structure indicated in the code skeleton. This will be used by the autograder to run the tests that will assign you a grade. You will only see a feedback for some of the tests as indiced in each problem above. All `print` operations from your side will be available on the grader so you can use that for debugging. However, try to remove them from the final submission of your work. The autograder will take a few seconds to run your code.

Regrading: Although unlikely due to the automation, if you run into grading issues (example: you think your solution is correct but does not passes the autograder tests), please request a regrading directly via Gradescope.

Autograder software issues Please, report any **software issues** (other than regrading) to `https://forms.gle/GXj9ELSq6rBnvJJq9` If you don't see any output from your submission it is not that the grader broke, but that the code uploaded has some issue. For instance, a library not installed in the Gradescope Grader is imported (see note above), or the name of your script is not formatted as indicated in the instructions at the top of this document.