CS-410 Text Information Systems – Spring 2026

HOMEWORK 1

Handed out: Feb 4, 2026 **Due:** Feb 20, 2026 at 23H59 CT

Deadline to post questions on Campuswire: 2 days before the due date, i.e., Feb 18 (You can post questions after this deadline, but a reply can take time to be posted, depending on the complexity of your issue, and thus it is unlikely it will help you to complete your homework on time). A question must be posted before the deadline to count towards class participation (extra points).

Submission through Gradescope (Problems 1, 2, and 3 provide immediate (but partial) feedback; Problems 4 and 5 are graded after the deadline). You need to ensure the code runs in the autograder for all the problems: upload the .py files with the names textretrieval.py (i.e., problems 1-3), 4_w2v.py (i.e., problem 4), and 5_optional.py (problem 5). Please, submit your work (Python script files) before the deadline (above)

GENERAL INSTRUCTIONS

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/gulli/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 will use PANDAS for processing and handling data.

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

PART I: PYTHON PROGRAMMING - IMMEDIATE NONCOMPREHENSIVE FEEDBACK (SUBMISSION VIA GRADESCOPE – File: textretrieval.py)

In Part I, you will receive immediate feedback on your submitted code. The list of items for which you will receive feedback is provided for each problem below. Since Part I provides immediate but incomplete feedback, you will have access to a skeleton/template code (on Canvas) to create your solution, which you will upload to Gradescope in the format (including the filename) indicated above. Please don't modify the class or function structure or signatures; i.e., don't change the parameters or default values. These are needed for the autograder to run your code. You can

code and test your solution locally, then upload it to the autograder (Gradescope) to test it. The autograder will not have the data available. You should code the data access yourself and upload only the script; please don't upload the data file. You don't need to install anything locally for the autograder; develop your code and upload it to Gradescope, which will run it. The template code includes a TODO list to guide you, but it's incomplete. You need to ensure you follow the instructions in this document.

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 collection. Sort them in decreasing order of frequency. Because you are using the `test.csv` file, the dataset in this implementation is small and manageable on 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 auto-grader will provide you with immediate feedback only for:

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

Other tests will run in the background and won't be available to the student, since they are run across various sections of the class. You may be eligible for deductions based on the 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"

The auto-grader will provide you with immediate feedback only for:

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

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"

The auto-grader will provide you with immediate feedback only for:

`3.1) computation of IDF`
`3.2) text2Tfidf representation`

A hidden test will evaluate the remaining tests. A manual grading component will also be performed for problems in Part I. If you used a library call instead of implementing the TF-IDF (representation and scoring) functions, you would get only 10% or 0% (90% or 100% deduction) of the score depending on the level of effort.

Write your code for all functions in a Python file and upload it to Gradescope. Ensure you rename the file before you upload it: `textretrieval.py` A template is available to guide you. The template has a reference TODO list. This list is non-comprehensive. Grading will be done based on completing the tasks listed in this instruction document.

PART II: PYTHON PROGRAMMING (SUBMISSION VIA GRADESCOPE – File: 4_w2v.py)

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 use

a library instead. 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 (score based on word vectors). 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"

Write your code in a Python file and upload it to Gradescope. Name this file: `4_w2v.py`.

PART III: PYTHON PROGRAMMING - EXTRA CREDIT
(SUBMISSION VIA GRADESCOPE – File: 5_optional.py)

5 (EXTRA CREDIT - OPTIONAL) Document Relevance with Vector Space (10 points)

a) (5 points) Okapi-BM25 with pivoted length normalization: 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.

b) (5 points) Pivoted length normalization VSM: As in Part I, implement a solution to Problem 3, but replace the whole formula with the Pivoted Length Normalization VSM retrieval model.

$$f(q,d) = \sum_{w \in q \cap d} c(w,q) \cdot \frac{\ln\big(1 + \ln\big[1 + c(w,d)\big]\big)}{1 - b + b\frac{|d|}{avdl}} \log \frac{M+1}{df(w)}$$

Write your code for both functions in a Python file and upload it to Gradescope. Name this file: `5_optional.py`

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, or 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.

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 feedback for some of the tests as indicated 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 (only remove the print calls you use for debugging, not the print operations you must use, per the instructions above). The autograder will take a few seconds to run your code.

Plagiarism This is an individual assignment. No external use is allowed, neither from human nor AI sources, as indicated in the course syllabus. The percentage of similarity of your homework with other students' or external sources will be used for grading purposes. If the similarity is detected by Gradescope or Turnitin Antiplagiarism software, the submission will be subject to penalties (such as a zero in the homework or the full homework component of the final course grade and include a FAIR report). A full description of potential penalties is listed in the course syllabus.

Regrading: Although unlikely due to the automation, if you run into grading issues (example: you think your solution is correct but does not pass 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.