CSC 483/583: Assignment #2 (75 pts)

Due by 11:59 P.M., September 26
(problems 1 – 7 in Gradescope; upload code for problem 8 to GitHub Classroom)

Because the credit for graduate students adds to more than 75 points, graduate students' grades will be normalized at the end to be out of 75. For example, if a graduate student obtains 80 points on this assignment, her final grade will be $80 \times 75/95 = 63.2$. Undergraduate students do not have to solve the problems marked "grad students only." If they do, their grades will not be normalized but will be capped at 75. For example, if an undergraduate student obtains 80 points on this project (by getting some credit on the "grad students only" problem), her final grade will be 75.

Note only problem 7 requires coding.

Problem 1 (5 points)

Suggest what tokenization and normalized form(s) should be used for these words (including the word itself as a possibility). Justify your decision.

- 'Cos
- Shi'ite
- cont'd
- Hawai'i
- O'Rourke
- ain't
- me@privacy.net
- <html> Some text </html>

Type your answer for problem 1 here:

Problem 2 (5 points)

Assume a biword index. Give an example of a document (could be a made up paragraph) which will be returned for a query of "New York University" but is actually a false positive which should not be returned.

Type your answer for problem 2 here:

Problem 3 (15 points)

Shown below is a portion of a positional index in the format: term: doc1: ⟨position1, position2, ...⟩; doc2: ⟨position1, position2, ...⟩; etc.

angels: 2: ⟨36,174,252,651⟩; 4: ⟨12,22,102,432⟩; 7: ⟨17⟩;

fools: 2: ⟨1,17,74,222⟩; 4: ⟨8,78,108,458⟩; 7: ⟨3,13,23,193⟩;

fear: 2: ⟨87,704,722,901⟩; 4: ⟨13,43,113,433⟩; 7: ⟨18,328,528⟩;

in: 2: ⟨3,37,76,444,851⟩; 4: ⟨10,20,110,470,500⟩; 7: ⟨5,15,25,195⟩;

rush: 2: ⟨2,66,194,321,702⟩; 4: ⟨9,69,149,429,569⟩; 7: ⟨4,14,404⟩;

to: 2: ⟨47,86,234,999⟩; 4: ⟨14,24,774,944⟩; 7: ⟨199,319,599,709⟩;

tread: 2: ⟨57,94,333⟩; 4: ⟨15,35,155⟩; 7: ⟨20,320⟩;

where: 2: ⟨67,124,393,1001⟩; 4: ⟨11,41,101,421,431⟩; 7: ⟨16,36,736⟩;

Which document(s) if any match each of the following queries, where each expression within quotes is a phrase query?

1. "fools rush in"
2. "fools rush in" AND "angels fear to tread"

Type your answer for problem 3 here:

Problem 4 (5 points)

Write down the entries in the permuterm index dictionary that are generated by the term "pandemic".

Type your answer for problem 4 here:

Problem 5 (15 points)

Compute the edit distance between "paris" and "arid". What are the N (rows) and M (columns) dimensions of the edit distance matrix? Write down the N $\times$ M array of distances between all prefixes as computed by the edit distance algorithm in Figure 3.5 in IIR. For each cell in the matrix, use the four-number representation to keep track of your intermediate results.

Type your answer for problem 5 here:

Problem 6 (10 points) GRAD STUDENTS ONLY

Consider the following fragment of a positional index with the format:

word: document: ⟨position, position, ...⟩; document: ⟨position, ...⟩
...

Gates: 1: ⟨3⟩; 2: ⟨6⟩; 3: ⟨2,17⟩; 4: ⟨1⟩;
IBM: 4: ⟨3⟩; 7: ⟨14⟩;
Microsoft: 1: ⟨1⟩; 2: ⟨1,21⟩; 3: ⟨3⟩; 5: ⟨16,22,51⟩;

The $/k$ operator, word1 $/k$ word2 finds occurrences of word1 within k words of word2 (on either side), where k is a positive integer argument. Thus $k = 1$ demands that word1 be adjacent to word2.

1. Indicate which set of documents that satisfy the query Gates /2 Microsoft.
2. Show each set of values for k for which the query Gates $/k$ Microsoft returns a different set of documents as the answer.

Type your answer for problem 6 here:

Problem 7 (5 points)

Artificial intelligence (AI) and automation in general clearly improve the quality of our lives (think Google Scholar). However, in many cases, they also eliminate jobs (e.g., the self-driving car impacts the livelihood of taxi drivers). Most often, these negative side effects impact the people least prepared to recover. If you were a policy maker, how would you address this problem? Please describe your solution and explain one of its advantages, and one drawback.

Type your answer for problem 7 here:

Problem 8 (25 points undergrads, 35 grads)

Implement an inverted index that supports proximity search. Your program must take in one file containing one document per line, in a format similar to the one from Assignment #1 (see that assignment for a detailed description of the format). For example, you can use the file below to test your code:

```
Doc1 breakthrough drug for schizophrenia
Doc2 new schizophrenia drug
Doc3 new drug for treatment of schizophrenia
Doc4 new hopes for schizophrenia patients
```

To code this problem, you can use Python or Java (or another JVM language such as Scala). You can use data structures available in your programming language of choice, e.g., dictionaries in Python or hash maps in Java/Scala, but you are **not** allowed to use open-source code that implements inverted indices, such as Lucene. You have to implement the inverted index and corresponding search operations from scratch.

The code submitted **must** pass the unit tests in the GitHub Classroom repository to be considered for grading. Please see the submission instructions for GitHub Classroom at the end of this problem.

Please implement the following:

1. **(25 points)** Construct a positional index and add support for Boolean proximity queries using the $/k$ operator. That is, word1 $/k$ word2 finds occurrences of word1 within k words of word2 (on either side), where k is a positive integer argument. Hint: use the algorithm from Figure 2.12 in the IIR textbook. What does your code return for the file above and the query: schizophrenia /2 drug? How about schizophrenia /4 drug?
2. **(GRAD STUDENTS ONLY: 10 points)** Modify the above algorithm to be directional. That is, the query word1 $/k$ word2 must return occurrences of word1 strictly **before** word2, within k words. What does your code return for the file above and the query: schizophrenia /2 drug?

For all these questions, please make sure **you return the documents in lexicographical order**. For example, if your query returns two documents with positional information, your output should be: <`Doc1`, 3, 4>, <`Doc3`, 4, 5>, where the numbers indicate the positions of the two keywords inside the documents. However, the internal implementation of both postings lists and positional information should rely on **integers rather than strings for both doc ids and document positions**. It is your responsibility to map the integers used for document ids back to the document names required by the unit tests.

You will implement and submit this problem using GitHub Classroom:

- If you program in Python, click on this link and follow the instructions on the screen: TO BE POSTED SOON
- If you program in Java, click on this link and follow the instructions on the screen: TO BE POSTED SOON

Note: if you are an undergraduate student, you do not have to address the last question. Leave the code for this question as is.

Very important note: **make sure the unit tests in your project pass on GitHub, after you submit your pull request!** If they do not, you will lose 50% of the credit for this problem, i.e., 15 points for graduate students, and 10 points for undergraduates.