Homework #2

CS 547/DS 547, Fall 2024

---

100 points total [6% of your final grade]

Due: Sept 30, 2024 by 11:59pm
[no submission will be accepted after Oct 3, 2024 at 11:59pm]

Delivery: Submit via Canvas

Programming [75 points]:
Handling wildcard queries through a binary tree based permuterm index
In this assignment, you are provided with a binary tree based permuterm index. This index includes permuterm variations of all terms within the documents. Your program should handle not only a regular query (e.g., 'hello'), but also a wildcard query (e.g., 'hel*o'), including multiple terms (e.g., 'hello world' and 'hel*o world'). By downloading hw2.zip and decompressing it; you should find three python files and five sample document files.

a) cs547.py - Just like HW1, this helper class will be used to represent a student's identifying information. Any assignments without an instantiated student object of type Student will not be graded. You do NOT need to modify this file.

b) binarytree.py: An implementation of a binary tree. This class is called in hw2.py to build a binary tree index and to search terms. The existed code populates this tree with permuterm variations all linking to lists of documents containing those entries. You do NOT need to modify this file itself.

c) hw2.py: This is where you will fill out two functions: wildcard_search_or(...) and wildcard_search_and(...). You should notice that five functions are already filled out: crawl_tree(...), permute(...), rotate(...), index_dir(...), tokenize(...).

What you already have:

- crawl_tree(node, term): This function crawls a binary tree looking for permuterm matches. In the function, whether term is a wildcard query or a regular query is checked. This function can be called in itself.

- permute(self, term): This function generates and returns a list of permuterm variations for the given term. For example, if term is 'hello', the list of its permuterm variations consists of 'hello$', 'ello$h', 'llo$he', 'lo$hel', 'o$hell', '$hello'.

- rotate(self, term): This function rotates a wildcard term to generate a search token, if the term includes a single '*'. Otherwise, return the input term. For example, if term is ‘hel*o', this function will return 'o$hel*'. If term is 'hello', it will return 'hello$'.

- index_dir(base_path): This function crawls through a directory of text files and generate a permuterm index of the contents using a binary tree. Tokenize() and permute() will be called.

- tokenize(self, text, is_search=False): This function converts a string of terms into a list of valid tokens. If is_search is False, a valid token will consist of only English alphabet characters (a-z,

A-Z). Otherwise, a valid token will consist of only English alphabet characters (a-z, A-Z) and '*' character. Also, this function converts all letters to lower case.

What you need to fill out:

- wildcard_search_or(self, text), wildcard_search_and(self, text): These two functions search for the terms in "text" in our index. Text can be a single term or multiple terms with/without a single '*'. Return a list of filenames with appropriate results. tokenize(), crawl_tree() and _rotate() will be called.

We have provided a sample set of five documents. Feel free to test your code over these five documents. You should be able to manually check whether your code is doing what you think it should. Once you submit your final code, we will evaluate it by constructing an index over our own set of 1,000 documents in a single folder and issuing several queries (single term and multiple terms with/without a single '*' symbol). You do NOT need to consider a wildcard query including multiple '*' in this assignment (e.g. he*l*). We have provided a sample set of five documents. Feel free to test your code over these five documents. You should be able to manually check whether your code is doing what you think it should. Once you submit your final code, we will evaluate it by constructing an index over our own set of 1,000 documents and issuing several queries (single term and multiple terms).

Short-Answer Questions [25 points]:

1. Investigate two search engines (e.g., Google and Bing), and figure out what tricks they are using by explaining your testing method. (15%)

(a) Do both search engines use stemming?

(b) Do both search engines filter stop words?

2. For a particular search query, your IR system returns 14 relevant documents and 16 irrelevant documents. There are a total of 80 relevant documents in the collection. (10%)

(a) What is the precision of the system on this search?

(b) What is the recall of the system on this search?

What to turn in:

- Rename hw2.py to hw2_firstname_lastname.py (e.g., hw2_steve_jobs.py) and submit to Canvas your **hw2.py file** and a **document file** consisting of answers of the question.
- This is an individual assignment, but you may discuss general strategies and approaches with other members of the class (refer to the syllabus for details of the homework collaboration policy). At the top of hw1.py you will see a list of COLLABORATORS. Please fill this out with the names of classmates you consulted and the nature of your discussion.