Assignment 7 YEAH Hours

Elina Thadhani

Warmup: common(lst1, lst2)

Goal: find the number of common elements in two lists and return a list of common elements between the two lists

Examples:

[‘a’, ‘b’, ‘c’, ‘d’] and [‘a’, ‘b’, ‘e’, ‘f’] -----> [‘a’, ‘b’]

[‘a’, ‘b’, ‘c’, ‘d’] and [‘e’, ‘f’] -----> []

Warmup: common(lst1, lst2)

Goal: find the number of common elements in two lists and return a list of common elements between the two lists

Tactic:

- Go through the lists and “save” only elements that appear in both lists
- Be careful to not “save” an element more than once if it appears in each list more than once

Bajillion: Overview

1. Building an inverted index
2. Building a filename-title mapping
3. Doing conjunctive searches

Bajillion: Inverted Index

create_index(filenames, index, file_titles)

This function is passed the following information:

- **`filenames`**: this is a list of file names (strings) that you'll use in building an index.
- **`index`**: this is a dictionary representing the index that you will need to build up. When your function is called, it will be passed an empty dictionary ({}) for index. Since dictionaries are mutable types, any changes your function makes to the parameter index will persist after your function completes.
- **`file_titles`**: this is a dictionary where the keys are file names (strings) and the values are the titles of the articles in each file (which are also strings). We'll explain the details of this parameter later in this handout. When your function is called, it will be passed an empty dictionary ({}) for file_titles and your function will add entries to this dictionary as appropriate.

Bajillion: Inverted Index

What is a term?

- Terms are seperated from each other in text by spaces or newline (return) characters.
- Terms should have all their letters converted to lowercase.
- Terms should have all punctuation symbols stripped off from their beginning and end. (Punctuation characters in the middle of a term are fine and should not be removed).

Bajillion: Inverted Index

Tactic:

- Read through a file, and find the terms
- Add the terms to the index dictionary where term is the key and value is a list of filenames where that term appears

Bajillion: Search one term

Goal: Use your index to search for a single term

- index: this is the index produced by your create_index function
- query: this is a string representing the user's query. All the letters in this string are guaranteed to be lowercase (the starter code we provide uses the lower function to create a lowercase query string that is passed to this function).

- return a list of the names of the files that contain all of the terms in the given query

Bajillion: Search

Goal: Given an index and query search return a list of all documents that have all terms in the query

Tactic:

- Split the query into individual terms
- Look up each term
- Make sure to calculate overlap!

Bajillion: Search

Overap: if a query has more than one term in it, return the list of files that have ALL terms

Tactic:

- find the documents that contain the first term and simply maintain a running intersection with the following terms …. Where have you looked for overlapping elements before?