CSE 674: Advanced Data Structures and Algorithms

Due on Tuesday, March 21 by midnight.

**Assignment #3:** Design and Implementation of a Document Handling System

In this project you are required to design and implement a Document Handling System that supports misspelled word detection and word suggestion, automatic competition of a given prefix, and efficient document retrieval. You are required to think about the data structures that you need to use and how to implement them. You are going to be provided with a text file that contains a large number of English words and a set of files to be used for document retrieval part.

System Requirements

The Document Handling System is supposed to implement following features:

1. Spell Check and Word Suggestion: reading a text file, your system is required to pinpoint all the misspelled words in the file. A word is considered to be misspelled if it is not found in the dictionary. You will use the dictionary.txt file for this purpose. You need to think about possible data structures that you can use for a fast look up and which ones will support the implementation of all the requirements of the system. Once you determine that a word is misspelled, you are required to suggest up to 10 words as spelling suggestion for the misspelled word.
2. Auto-complete: Your system interface should provide a function that returns a set of words that all start with a given prefix. You are going to be using the same data as in feature 1 and this time try to achieve a fast retrieval of the words that have that prefix.
3. Document Retrieval: Given a set of files and key words, you system will print the names of the files that contain the given keywords in an efficient manner. Your system will also detect any possible misspellings and continue the retrieval process after automatically replacing the misspelled key word with the first one amongst the system’s suggestions from the feature one.

Design and Implementation

1. Spell Check and Word Suggestion: Trie is the best option to implement this feature. You need to create the trie and insert all dictionary words into it from the dictionary file. Hashtables could also be a very good and efficient option but they will not be very practical with auto-complete feature. Once the trie has been created, you need to read a text file and find all misspelled words in it. In addition, you are also required to suggest correct spelling for each one of these words.

*Vector<string> suggestions*

*Queue<string> words*

*Words.enqueue(word)*

*While words.size>0 && suggestions.size<10*

*String s=words.Dequeu()*

*Generate 1 character away words of s using substitution, insertion, and deletion*

*Words.Enqueu(all of these words)*

*Insert the actual words into suggestions*

1. Auto-complete: For this you need to locate the node that represent the given prefix and do a depth first search traversal for the subtree rooted from that node. These are the actual words that start with that prefix.
2. Document Retrieval: For this part you need to first create what is called an inverted index. An inverted index can be implemented as a hashtable where keys are strings that represent words in documents and the values for each key are the document names that contain that word. So you need to first read the entire set of files and create this hashtable. So if a word is included in more than one file, the value for that key will be a list of all files that include that word. You need to read each file, tokenize the words, eliminate any punctuations and put each word into the hashtable and repeat the same thing for the second file.

Once you have your inverted index, you can run queries on it very easily. For this you need to find the list of all file names that include each query word and print the intersection of these lists. Like sais before, you need to verify that the query words are spelled correctly before you search for the files, otherwise use the first suggestion for the misspelled query words and list the file names after printing a statement such as “did you mean this: *same key words except the misspelled ones are replaced with the first suggestions*.”

*Note: Required files are going to be posted on blackboard. 5 points off per day late.*

*dictionary.txt includes most common 20000 English words. You need to use this to detect misspelled words. The FileSet.rar includes a relatively large number of text files that has some random content. When you create your inverted index you should probably eliminate all punctuations. Lastly, the TobeSpellChecked.txt file is just an example for you to test your spellchecker. You are required to find all misspelled words in this file and suggest correct spellings. I do not know if it contains any misspelled words right now, but you can throw in a couple just to make sure that you detect them.*