Trie-based Word Completion

Cpt S 223 Homework Assignment

by Evan Olds

Submission Instructions:

Submit source code (zipped) to Angel BEFORE the due date/time. If the Angel submission is not working, then submit to TA via email BEFORE the due date/time. Optional: Include a readme.txt file in the zip with any relevant information that you want the grader to be aware of.

Assignment Instructions:

Read all the instructions *carefully* before you write any code.

First note that there is not a Visual Studio solution file included in the zip. You will be creating a project from scratch.

Create an application that takes two command-line arguments. The first is the filename of a dictionary file. The second is the file name of a list of words that you need to "complete". These files are described below.

Dictionary File

This is a list of words that you will use to build your trie. There is one word per line. The words list was obtained from http://www-01.sil.org/linguistics/wordlists/english/ but the line endings were normalized so make sure you use the wordsEn.txt file from the zip and do not re-download it from the web or you may have problems. The web link is just to cite the source.

Input File

This will be a short list of words, one per line, that you need to use as prefixes for a dictionary lookup. For each word in this file, your application will get a list of all words in the dictionary that start with that prefix and display them (one per line).

Output produced by your application

Your application may be graded by an automated grading program so make sure you match the required output format.

1. The first line of your output should have your name and ID number

2. The second line is a message that is displayed after building the trie from the dictionary file. It should say how many lines were loaded from the file and should look something like “Built trie from 109582 dictionary words"

Remaining lines are words from the auto-completion of the words you load from the second input file. See the included sample input and output files for an example of what your application will produce.

That is all for the assignment requirements, remaining content is just helpful information.

Loading lines from files

You application code may start something like this:

```
int main(int argc, char* argv[])
{
	if (argc < 3)
	{
		cout << "Missing required input file names\n";
		return -1;
	}

	string line;
	std::ifstream dictionary(argv[1]);
	std::ifstream lines(argv[2]);

	if (!dictionary.is_open())
	{
		cout << "ERROR: Could not open file " << argv[1] << endl;
		return -2;
	}
	if (!lines.is_open())
	{
		cout << "ERROR: Could not open file " << argv[2] << endl;
		return -2;
	}

	cout << "(student name and ID will be here on 1 line)\n”;
```

After you’ve loaded the files, you can use the getline function to read lines. There are a variety of ways to load the files. Use whatever method suits you best, the above way of using ifstreams is just one option.