Myles Maxfield
Summer Mentorship – it.com

Project Proposal

Objective: My project will be to create a third-generation spell checker for it.com's search engine. I hope to create an integrated spell checker that will transparently check each word in each query entered into the search engine.

Justification: A spell checker is needed because if the search query is spelled wrong, there will be no pertinent results found. Searching a database for an unrelated topic will not return useful results. We can help the user include useful search terms.

The spell checker that is currently used by it.com has many problems. First, it does not detect if two letters in the search query are switched. It would detect a switch of two letters as two errors, and not one. Also, the spelling suggestions it gives are less than stellar. For instance, I typed in the search "appel" and the best match it found was "appear," and not "apple." This best match is a verb, which would not be searched on much. The current spell checker also does not take similar-sounding syllables into account when searching for a good match. For instance, I searched for "phear" and the best match it found was "hear" and not "fear".

Description: First, I will research other algorithms used for string matching. These algorithms are not simply finding the fastest or most memory efficient way to match two strings exactly. Instead these algorithms are known as the K-Mismatch problem. These algorithms match two strings with K differences, or mismatches. That means that there are K times when a letter has been swapped in the two strings, or inserted into one string, or deleted in a string.

Once I am comfortable with these string matching algorithms, I will either move on to spell checking algorithms, or try to adapt the string matching algorithms to function as a spell checker. I am mostly concerned with the nearest-neighbor algorithms that will find me a correctly spelled word that would be closest to the word entered. I also will create new data structures for the dictionary, such as suffix trees. This facilitates fast searching of a word in a dictionary.

I do not have to simply create a spell checker, but I must create numerous other programs first about string matching, before I can even attempt to write a spell checker. I must learn to create data structures that help store a dictionary.

Limitations: The spell checker only checks the spelling of the query, it does actually provide any searching functionality. It does not provide any functionality for the actual searching, and only helps the user type what they mean to type. Also, my algorithm cannot be perfect because a computer can never know what a user is thinking. My biggest constraint is time. I might not have enough time to complete a professional-grade spell checker. This time constraint poses as a larger problem the more I create intermediate algorithms, such as the string matching algorithms. This spell checker will be implemented only to provide one line of text in the actual search engine, asking if the user meant another search term.

Background Research:

Allison, Lloyd. "Suffix Trees." Clayton School of Information Technology. 2000. 30 June 2006 <http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Tree/Suffix/>.

Kiefer, Stefan. "The Least Common Ancestor Problem." JASS 2003. 30 Jan. 2003. 2 July 2006 <http://www14.informatik.tu-muenchen.de/konferenzen/Jass03/presentations/kiefer.pdf>.

Landau, G., and U. Vishkin. "Fast Parallel and Serial Approximate String Matching." Journal of Algorithms 10 (1989): 158-169.

Nelson, Mark. Fast String Searching With Suffix Trees. Aug. 1996. 30 June 2006 <http://www.dogma.net/markn/articles/suffixt/suffixt.htm>.

"Suffix Tree." Wikipedia. 24 June 2006. 29 June 2006 <http://en.wikipedia.org/wiki/Suffix_tree>.