Gabrielle Ehrlich && Steve Tricanowicz

Computer Science 51

2/27/10

Moogle Part 1 : Design Doc

Summary:

We plan to implement several small features to improve Moogle's functionality. These are optional case insensitivity, phrase searches, and spelling suggestions. We are going to have a button on the moogle page that allows the user to choose whether or not the search is case sensitive. When a search is performed, if the word is not in the dictionary, Moogle will suggest words that were close to it.

Non Goals:

We are not going to implement the solution to the Israeli Palestinian conflict, nor are we going to implement and end to World Hunger (in this version at least). We are also not going to implement page rank. We are also not going to implement a virus that takes over the grading control system, although this topic was hotly debated.

Scenarios:

Bob is searching through his computer files and wants to find a document that he wrote about french toast. Unfortunately, Bob has very bad grammar habits and tends to capitalize things at random and doesn't remember if he wrote FrenCH tOast or fREnCh ToaSt. By turning of case sensitivity he will be able to find the file without many repeated searches.

Joe is Bob's best friend and he shares some of his oddities, but Joe's problem is that he wrote a bunch of files about Super Conductors. He remembers that in the file he wants, however he at one point wrote SUPER Conductor with super in all caps. By turning on case sensitivity he will also be able to find his file without sorting through tons of files.

Sarah is a very bad typist, so she always misses keys or hits the wrong key when she is typing, so our spelling suggestions will help her get to the query she intended.

Details:

We are going to have to decide whether Moogle should suggest the single closest word, to the misspelled word, or a list or similar words. We will also have to decide whether to weight certain edit distances more than others.

Changes to Data Structures:

We aren't going to need to change the data structures because our functions will only work with the words passed into moogle, and change what is searched on rather than how the search is done. We also

are going to order suggestions based on number of results, and while we are populating the list, we will keep the searches around, so when was is selected, it won't have to research.

New Algorithms:

For algorithms we will need a generate permutations algorithm, whose signature will be:

(* create permutations and calculates the edit distance of a list of words *)

let rec permutate (s : string) : string list =

New Data Structures:

We don't believe we will need a new data structure because everything we are doing can be done with preexisting structures. Depending on the implementation of permutations, we may wish to give priority to those with low keyboard distance (i.e. keystrokes were not far away from what was expected on a normal QWERTY keyboard). To do this, we will probably need some sort of map to keep track of distances between keys.

Division of Labor:

Most of the work for the extension will be done together. For case insensitivity, it may be efficient to either delegate that to one person, or have one person implement the actual functionality and the other implement the optionality. As for suggestions, though, we will probably each write functions, depending on priority, and then look at them together. For a component as complex as this might prove to be, it is often more effective to have two sets of eyes looking at the code at once.