Keyboard auto-correction plugin

Daria Gorbunova, Vsevolod Averkov, Kirill Kulakov
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
{gorbunov, averkov, kulakov}@cs.petrsu.ru

***Abstract*—Autocorrection is a really valuable helper in modern life, especially in smartphones. Nowadays we can even write an essay with our phones and most of us use phones to text friends. It helps us to avoid most errors that appear in ours texts due to the lack of grammar knowledge or rush. The article presents a software tool for text autocorrection. The application can correct your errors while you are texting.**

The article presents a solution based on the analysis of the input word. The algorithm continuously checks the tree for the closest possibly correct word to the input word and returns it to the user. Plugin was created for the Aurora mobile operating system (Sailfish OS) in C ++ (Qt creator).

There are a lot of approaches to autocorrection, but the dircitonary approach has the most amount of advantages. This method allows the smallest number of errors. Moreover, only the dictionary method allows implementing in the auto-correction program the advancement of alternatives to the word being verified and the adoption of one of them as an automatic correction, which is necessary and can be used for auto-correction for the virtual keyboard (since the post-control of automatically corrected places by the user is provided).

Determining the possible correct word is a difficult task, we can choose the closest word to the input word in the dictionary, but we should be careful to decide which algorithm to use to realize every word check because the simple sorting will take a lot of time and memory. Our solution to that problem is using trees as a structure for the dictionary. Because of them the process of spellcheck will be faster due to the lack of need to check every single word in most cases. The most fitting kind of trees for this task is Burkhard Keller Tree.

It is also useful to observe the Livenstein's distance. This distance forms a metric space. And metric space is any relation that meets some basic criteria:

1) $D(x, y) = 0 <=> x = y$ if the distance between x and y is zero, then $x = y$
2) $D(x, y) = D(y, x)$ the distance from x to y is equal to the distance from y to x
3) $D(x, y) + D(y, z) \geqslant D(x, z)$ This is called a triangle inequality. The path from x to z should not be longer than any path passing through another intermediate point.
   $D(x, y)$ - integer discrete metric.

Let's say we have parameters: $dist$ -is the resulting distance Str-the string we use in our search. n - is the maximum distance that Str can be from the request and still return. Suppose we take an arbitrary string and compare it with Str. Since we know that the triangle inequality holds, all our results should have the largest distance $dist$+n. The overall architecture is shown in the Figure 1.

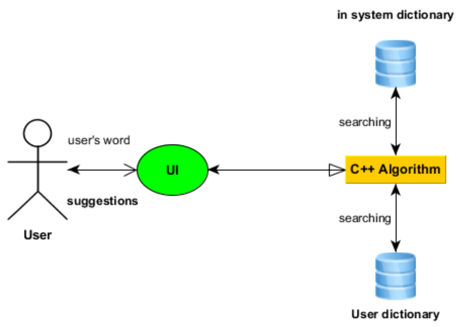

Fig. 1. Arcitecture of spelling application

The application works this way. The user begins to enter a word and similar words begin to be displayed to him. If a person is mistaken, the word will be corrected.

In the current stage of this progect four possible outcomes are possible for the user: there are no possible variant for correct word (see Fig. 2), there is one possible variant for correct word (see Fig. 3) there are two possible variants for correct word (see Fig. 4) and there are three possible variants for correct word(see Fig. 5).

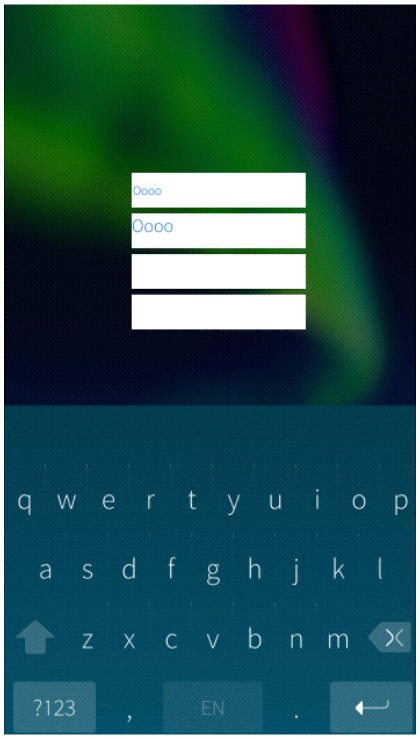

Fig. 2. No possible variant for correct word

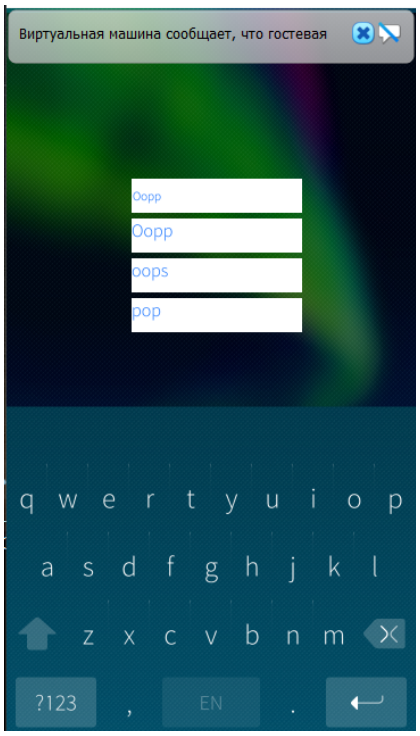

Fig. 4. Two possible variants for correct word

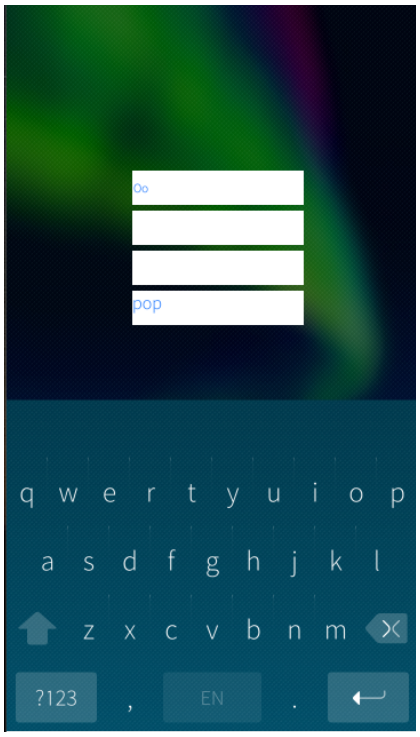

Fig. 3. One possible variant for correct word

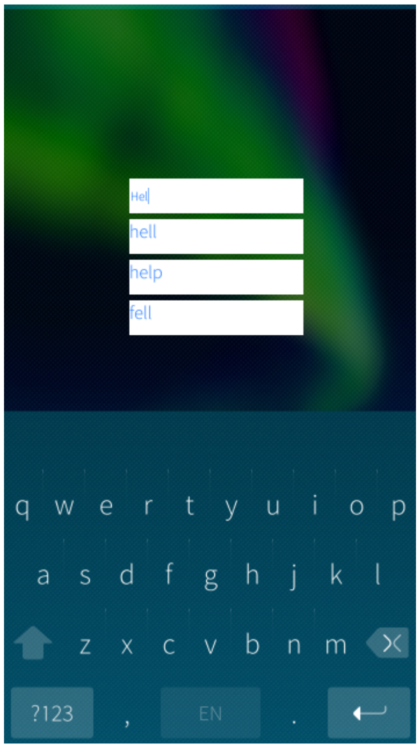

Fig. 5. Three possible variants for correct word