##################################################################################

The time taken to insert the whole 370k English word corpus was found to be 37 seconds which would make the app lag a lot during start up.

Even after the python generated .pyc compiled file…it was still 6 seconds.

So what I did was I pickled the trie object before hand and stored it in disk.

Then I found the pickle object was of 60mb which is large for a simple application. So I searched a bit and found a library bz2 that could compress the object. So now my file was 5 mb which was nice.

So whenever the .exe file is run it starts with loading this trie pickled object and does the operations as required. This now takes 1-2 seconds.

##################################################################################

I started out with giving out with using frequency of word used in case of spelling error. But this created some bias. Eg everyday I say : I need a latte. So the latte has high bias when compared to latest. So then instead of using frequency I used combination of Lavenshtein distance and frequency to predict the correct word.

##################################################################################

Now I found a problem with using such a large corpus. Eg . I wote a word bitsi . I want it to correct it to bits but instead it does it to bitsy because bitsy exists in this corpus, which is a very rare word that I don’t think so, much people uses. So I am thinking, that, instead of using search a large corpus. I should start small such as with few thousand of common words. And as the user starts using the app, it will learn the words that are not present in dictionary, so a truly personalized experience can be provided.

So I used a corpus of 10k words provided by mit.

This was a game changer. Now the 2-3 seconds that were taken on start-up were reduced to milliseconds, while providing better suggestions after some initial usage.

#################################################################################