Text autocompletion

Tom Rochette <tom.rochette@coreteks.org>

July 24, 2025 — daae079c

0.1 Context

0.2 Learned in this study

0.3 Things to explore

- Autocompletion based on more than word boundaries
- Is it possible to speed up the autocompletion process to make it more useful than being able to type rapidly?

1 Overview

2 Notes

- Words which require you to type them almost completely, such as "used" because "use" is more likely, do not benefit from autocompletion because typing the additional letter is as much work as accepting the autocompletion. However, since the target word ("used") is below the currently typed word, it would require more keystrokes to select (`go down`, `press enter` vs `type d`)
  - In other words, this means that a word list, ordered by most frequent to least frequent based on a document corpus, should drop all the words for which there is an edit distance of 1 with an word with lower index in the list
- Words for which the autocompletion can be done either by 1 keystroke or pressing enter should be configurable by the user (either they don't want to see them, or they want them so they can type or autocomplete)
- All the one letter words should be removed from the dictionary as they cannot benefit from autocompletion

2.1 Tests

- Given all my current articles, if I build a dictionary out of it where the keys are the words and the values are the associated count (number of instances in the corpus), and I use the current algorithm I have (a trie search which favors the most frequent first), I obtain about 15-17% of saved keystrokes on this same corpus. On an AGI article (in source form), I obtain a lower 6%
- Using Google 10k words, I get approximately 6% on this same AGI article
  - If I run this dictionary over my own set of articles, then I would obtain about 8% (whereas with my own dictionary I have 15-17%)
  - Tested on a random wikipedia article, both my own dictionary and Goole 10k word perform at approximately 4% saved keystrokes, which is rather interesting, given that my own vocabulary (and word frequencies) come from a much smaller corpus, yet both result in approximately the same keystrokes savings
- In my test data (my own articles), there are numerous elements which have high gain because they are extremely long and easy to recognize early on (such as urls)

2.2 Observations

- One of the downside of using autocompletion is that, unlike when you type, you need to look at the autocompletion window and actively attempt to make decisions, which is likely to strongly reduce the time saving we are looking for

3 See also

4 References

- https://medium.com/@curiousily/making-a-predictive-keyboard-using-recurrent-neural-networks-tensorflow-for-hackers-part-v-3f238d824218
- https://towardsdatascience.com/building-a-next-word-predictor-in-tensorflow-e7e681d4f03f