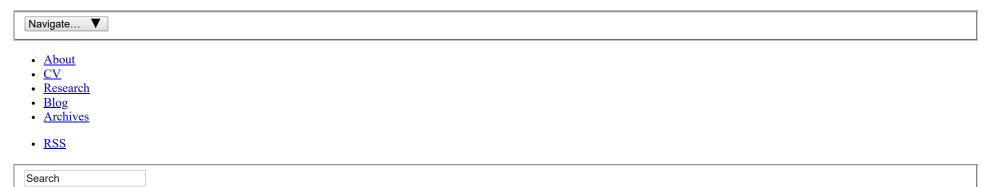
Publishable Stuff

Rasmus Bååth's Research Blog

• Twitter



Peter Norvig's Spell Checker in Two Lines of Base R

Dec 17th, 2014

Peter Norvig, the director of research at Google, wrote a nice essay on <u>How to Write a Spelling Corrector</u> a couple of years ago. That essay explains and implements a simple but effective spelling correction function in just 21 lines of Python. Highly recommended reading! I was wondering how many lines it would take to write something similar in base R. Turns out you can do it in (at least) two pretty obfuscated lines:

```
1 sorted_words <- names(sort(table(strsplit(tolower(paste(readLines("http://www.norvig.com/big.txt"), collapse = " ")), "[^a-z]+")), decreasing = TRUE))
2 correct <- function(word) { c(sorted_words[ adist(word, sorted_words) <= min(adist(word, sorted_words), 2)], word)[1] }</pre>
```

While not working exactly as Norvig's version it should result in similar spelling corrections:

```
1 correct("piese")
1 ## [1] "piece"
1 correct("ov")
1 ## [1] "of"
```

```
1 correct("cakke")
```

1## [1] "cake"



So let's deobfuscate the two-liner slightly (however, the code below might not make sense if you don't read Norvig's essay first):

```
1 # Read in big.txt, a 6.5 mb collection of different English texts.
2 raw text <- paste(readLines("http://www.norvig.com/big.txt"), collapse = " ")</pre>
3 # Make the text lowercase and split it up creating a huge vector of word tokens.
4 split text <- strsplit(tolower(raw text), "[^a-z]+")
5 # Count the number of different type of words.
6 word count <- table(split text)</pre>
7 # Sort the words and create an ordered vector with the most common type of words first.
8 sorted words <- names(sort(word count, decreasing = TRUE))</pre>
10 correct <- function(word) {</pre>
11 # Calculate the edit distance between the word and all other words in sorted words.
12 edit dist <- adist(word, sorted words)</pre>
# Calculate the minimum edit distance to find a word that exists in big.txt
14 # with a limit of two edits.
15 min edit dist <- min(edit dist, 2)</pre>
16 # Generate a vector with all words with this minimum edit distance.
17 # Since sorted words is ordered from most common to least common, the resulting
18 # vector will have the most common / probable match first.
19 proposals by prob <- c(sorted words[ edit dist <= min(edit dist, 2)])</pre>
20 # In case proposals by prob would be empty we append the word to be corrected...
   proposals by prob < \overline{c} (proposals by prob, word)
22 # ... and return the first / most probable word in the vector.
23 proposals by prob[1]
24 }
```

Some thoughts:

- The main reason for why the R version is so short is because base R includes the adist function. (A one line spell checker in R is indeed possible using the aspell function:)
- A second reason for why the R version is so short is that the many vectorized functions in R make it possible to do a lot of work in one line.
- Indeed, the horrible line creating the sorted words vector would be a perfect target for some magrittr magic.
- The R version does not solve the problem in exactly the same way as Norvig's code. He maintains the count of each word in the NWORDS variable in order to be able to extract the most probable matching word. This is not necessary in the R code, as we already have a sorted vector we know that the first item always will be the most probable. Still, I believe the two approaches result in the same spelling corrections (but prove me wrong:).
- There are links to implementations in many other languages at the bottom of <u>Norvig's essay</u>. Looking at <u>the Java version</u> reminds me of my dark Java past and madness like HashMap<Integer, String> candidates = new HashMap<Integer, String>();.

Posted by Rasmus Bååth Dec 17th, 2014 R

Tweet

« Eight Christmas Gift Ideas for the Statistically Interested Probable Points and Credible Intervals, Part 2: Decision Theory »

Comments

3 Comments Rasmus Bååth's webpage



C Recommend 3

У Tweet **f** Share

Sort by Best



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS (?)

Name



Peter Norvig • 4 years ago

Very nice, Rasmus. I think you are correct that this does basically the same thing as mine (except perhaps for breaking ties), and that the reasons why it is so concise are (1) the adist function, and (2) vector processing in R. Forgetting the word counts and just keeping the ordering is certainly justified, but I kept the counts because I was thinking ahead to extensions such as finding the most probable correction in the context of other words -- for that you would need probabilities, not just orderings.



Rasmus Arnling Bååth Mod → Peter Norvig • 4 years ago

Thanks! Yeah, it's mostly a hack and the python version is much clearer. Still, I think it's pretty cool that in just one row you can read the data from your server, tokenize the words, count the words **and** produce a sorted list. :)



icheng5 • 4 years ago

Cool! The two line version would be a lot more readable with some %>% usage :)

1 ^ Reply • Share >

ALSO ON RASMUS BÅÅTH'S WEBPAGE

Wrapping up Bayes@Lund 2015 - Publishable Stuff

2 comments • 4 years ago



Ase Innes-Ker — It was very nice. For me, who is still wading into this very slowly, the Pre-conference (or, Prior conference as my smart-aleck twitter buddies suggested) was

The Map of Romantic Kissing (and a critique)

3 comments • 4 years ago



Rasmus Arnling Bååth – Also I and my collegue Andrey Anikin spent a fair amount of hours on Wikipedia finding the coordinates of the different cultures so if you get ...

bayes.js: A Small Library for Doing MCMC in the Browser

2 comments • 3 years ago



aloctavodia — Nice! Another serious tool is PyMC3;-)

Posterior update of Bayes@Lund 2016

1 comment • 3 years ago



Umperior longer! Umberto - Thanks Rasmus and Ullrika for a great organization. Too bad I couldn't stay

Copyright © 2019 - Rasmus Bååth - Licensed under a Creative Commons Attribution 4.0 International License - Powered by Octopress