

12. Text Processing

By: Steve Krenzel

[Previous](#)

[Index](#)

[Next](#)

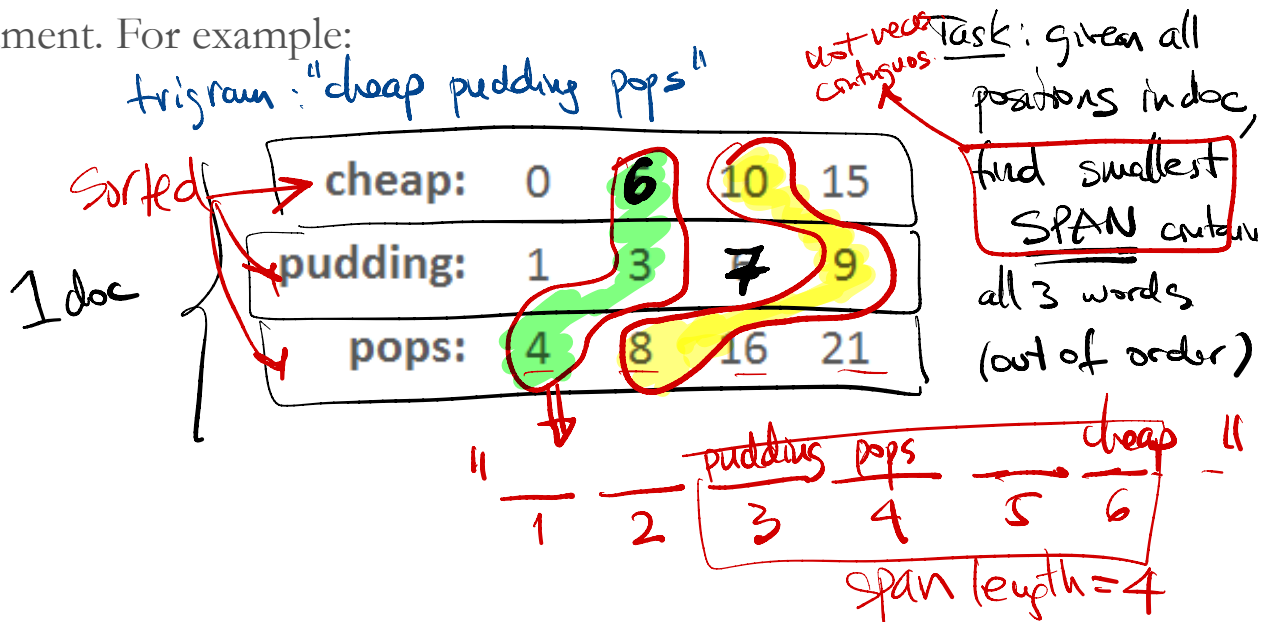
Finding Blurbs \Rightarrow proximity at runtime
using unigram positions.

Suppose you ran a search engine and someone searched for a three word phrase like "cheap pudding pops". Also suppose you've already solved the problem of finding relevant URLs. The problem you're solving now is showing a blurb from each of those websites on the results page. You want the shortest span of text in the document that contains all of the words you searched for.

I used to know the name of this algorithm but it escapes me at the moment and despite my attempts at googling and asking around various channels on IRC, no one seems to know. The algorithm I'm going to describe uses a sliding window technique across the positions of the words in the document. It runs in $O(N)$ time, where N is the number of occurrences in the document of the word with the most number of occurrences.

Here is the algorithm:

1. First, you need a list of all of the positions of the words searched for in the document. For example:



2. Then you keep track of our window, which initially starts at the beginning of each list:

	0	1	2	3	4
cheap:	0	5	10	15	
pudding:	1	3	6	9	
pops:	4	8	16	21	

Handwritten notes: $span = 4 - 0 + 1 = 5$, $span = 5 - 1 + 1 = 5$, $span = 5 - 3 + 1 = 3$ optimal

3. At this point our smallest known blurb is the window [0, 1, 4] which has a range of 4. To see if we can do any better, we move forward the window of the word with the smallest position, in the case it's the list for the word "cheap" because it's position in the window is 0. We get:

cheap:	0	5	10	15
pudding:	1	3	6	9
pops:	4	8	16	21

4. Our new window is [5, 1, 4] and has a range of 4 again. We repeat step 3, moving forward the window of the word with the smallest position. This time it is "pudding". We get:

cheap:	0	5	10	15
pudding:	1	3	6	9
pops:	4	8	16	21

Handwritten notes: $RT = O(n)$, $trigrams \Rightarrow 3n$, possible spans (all): n^3

Q: is it possible that the min-span is missed?

NO \Rightarrow ALG CORRECT

algorithm looks only at $3n$ spans out of n^3

5. Now our window is [5, 3, 4] and has a range of 2. This is our new smallest window. It is also the smallest window possible, so we could stop if we wanted to, but we won't because I want to show you what to do when you

• proof?

• ALG type? Greedy

get a window that hits the end of the list. If we keep repeating step 3, we eventually get:

****** I want All spans

size $\leq k$

ex. all spans size 5

\Rightarrow Spans of sizes 3, 4, 5

\rightarrow count the occurrences of n groups

• what if I want ALL min-spans?
 \downarrow

ALG produces all or it can miss some

cheap:	0	5	10	15
pudding:	1	3	6	9
pops:	4	8	16	21

size - #terms = \$bp

6. Our window with the smallest position is already at the end of its respective list, so we can't move it forward any more. In this case we simply go to the next smallest and get:

cheap:	0	5	10	15
pudding:	1	3	6	9
pops:	4	8	16	21

7. When we repeat the process again, all windows will be at the end of their respective lists. At this point we simply return the window that had the smallest range, which we calculated all the way back in step 5. There are no more windows to try.

If the shortest span is still too long for your blurb you'll have to find some other way to split it up, which we won't cover here. You could also use the shortest span as a metric in ranking relevant documents since you probably want documents that contain the words close together.

The algorithm as presented can be optimized in a number of ways, but complicates its implementation slightly. Doing so is left as an exercise.

[Here is some simple code](#) that implements this algorithm, it also has some basic

doctests included.

About the author

I'm Steve Krenzel, a software engineer and co-founder of [Thinkfuse](#). Contact me at steve@thinkfuse.com.



[Previous](#)

[Next](#)

©2010 Steve Krenzel