# Instant Search algorithm attempt

1. Grab search string.
2. Split into words on the space character.
3. Discard articles/superfluous words (The, it, a, etc).
4. Get 1-3 common synonyms for each search word.
5. With words and synonyms in hand, search database. Score each database entry by two metrics. User search words are scored more highly than synonyms.
   1. One metric is for if the entry has all the words in it.
      1. **Or the closest match, yea? i.e. No page has all words but a page does have ¾ or whatever. We should return that page.**
      2. Yeah definitely, it should read “One metric is for the percentage of search terms matched found on the page.” Or something.
   2. The other is for number of times the search words were found.
      1. **I am a little confused here: are you saying this will be done after “a” completes? If yes, then I think this is a great way to score pages.**
      2. I was thinking we could, during our search, record both metrics at the same time. So, a page with 4/4 search terms and the terms are mentioned 3 times will rank higher than a page with ¾ search terms and 45 search term mentions. We can always test out different variations.
6. Sort the data by metric 4.b, then 4.a.

The scoring part of the algorithm will likely need tweaking to find the right values.

This approach also doesn’t take order of search words into consideration.

**I think we need to account for order of words. Best example I can think of is the stark difference between “Red Bull” and “The bull is red” (I assume after stripping we would search “bull red”)**

**Unfortunately, I have not thought of a good way to go about this. Pesky associative property of Addition.**

**We are going to have to tweak how we discussed scoring the words.**

I’m still not convinced that the order will end up mattering, but let’s definitely do both: One where the order does matter and one where it doesn’t. I also don’t quite know how it would work haha, maybe just literally searching for the overall term “Red Bull”.

This approach does highly value just having all the search words in the data entry.

**What exactly do you mean by this?**

Just that a page with all the search terms would always be ranked above a page with some of the search terms. That may or may not be a good thing, I’m not sure.

Rough Example:

The user’s search string is “Bee sting”.

The string is split up into words “bee” and “sting”.

According to datamuse.com, synonyms for “bee” are “honeybee” and “wasp”. Synonyms for “sting” are “bite” and “twinge”.

A page is found that has multiple occurrences of both “bee” and “sting”. It scores a 2 (out of 2) in matching words and a high score in quantity of occurrences.

Another page is found that has multiple occurrences of “wasp” and “sting”. It scores a 1.5 (out of 2) in matching words and also a high score in quantity.

A page with just the word “bee” would score a 1.

**To expand algorithm:**

**Unless I am missing something, this is basically just showing how a search engine already works. I was under the understanding that we were going to show the user the words we replaced their words with that had successful hits. If I am correct:**

**After we search the page and find results: we should show the user which words had the highest relevance. I.E. if they search for “Research Chances” (I am working off the assumption our search data will be the catalog, but that can easily be changed) but our algorithm finds substantially more hits with “Research Opportunity”, are we going to display to the user the replacement? I would think it would be to terribly hard to implement.**

Yes, you’re right, the project is definitely asking that question. I was thinking that it’s kind of a UX issue and will be added on after the search algorithm (mostly) works. I think you’re onto something: that finding way more hits for a synonym is indicative of something worth returning/showing the user.

As for actually indicating which search terms were replaced, I guess we can just start off with bolding the synonym? Haven’t given it too much thought.