Instant Search Bar

Jeffrey Slentz

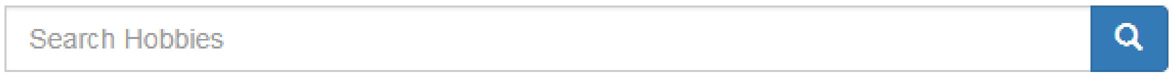
Introduction

The initial project's goal was to gauge user response to displaying similar but not equal words and queries to their search in a data set via a search bar. User's responses could be used to inform a good UI design that made it clear what was happening with their search query.

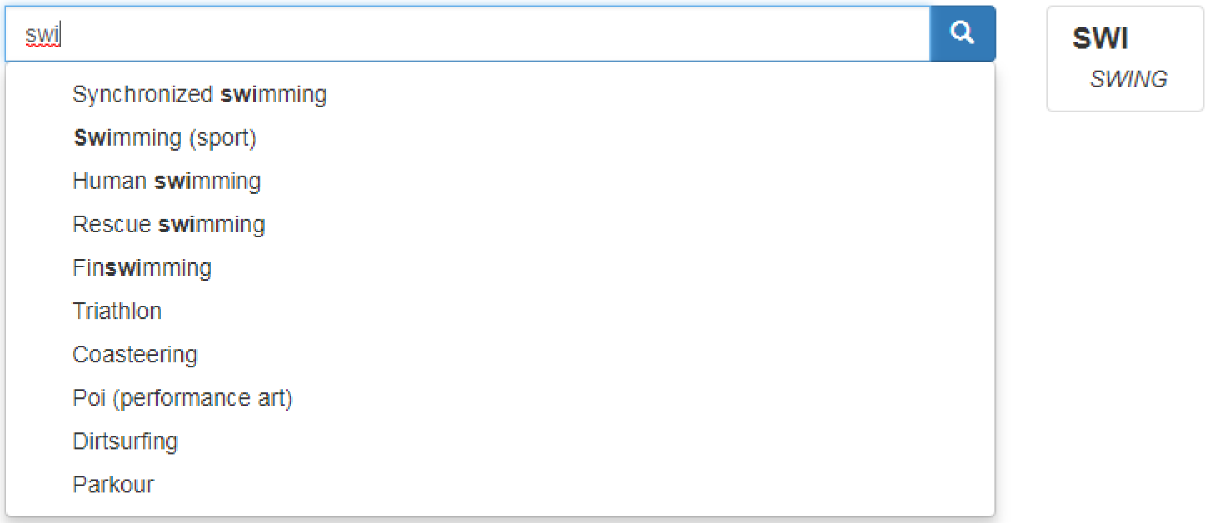
Design

The main functionality

The webpage initially consists of one element: the search bar.



As the user begins typing, the next two elements populate. The dropdown has the results and the suggestions box holds the words being used to score the results. In this case, an incomplete word swi is being autosuggested as swing.

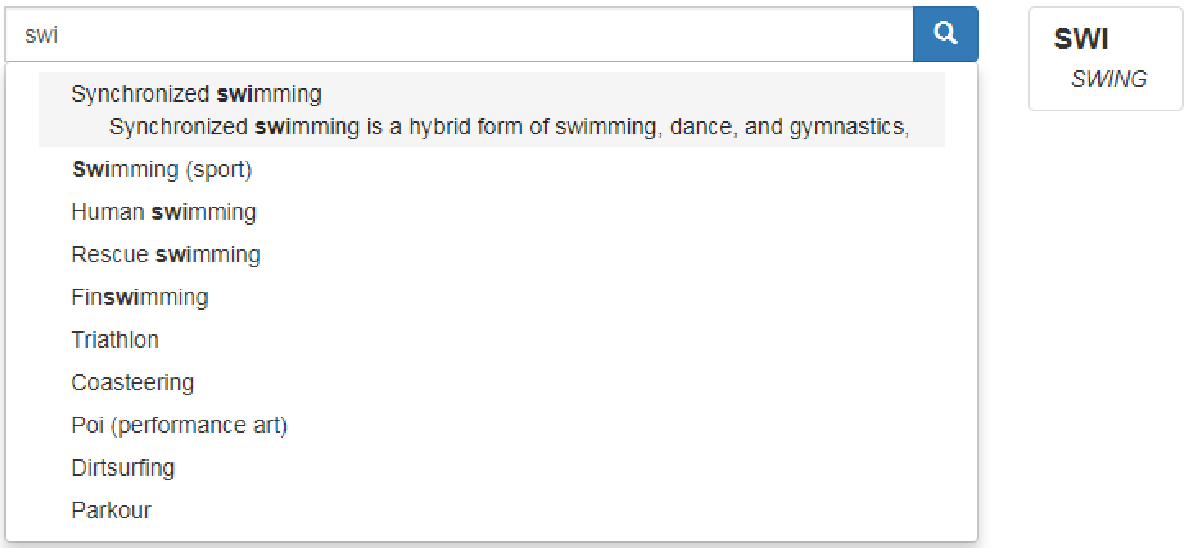


swi

Synchronized **swi**mming
Swimming (sport)
Human **swi**mming
Rescue **swi**mming
Fin**swi**mming
Triathlon
Coasteering
Poi (performance art)
Dirtsurfing
Parkour

SWI
SWING

If an element is hovered over, a snippet of the summary is shown.


swi

Synchronized **swi**mming
Synchronized **swi**mming is a hybrid form of swimming, dance, and gymnastics,

Swimming (sport)

Human **swi**mming

Rescue **swi**mming

Fin**swi**mming

Triathlon

Coasteering

Poi (performance art)

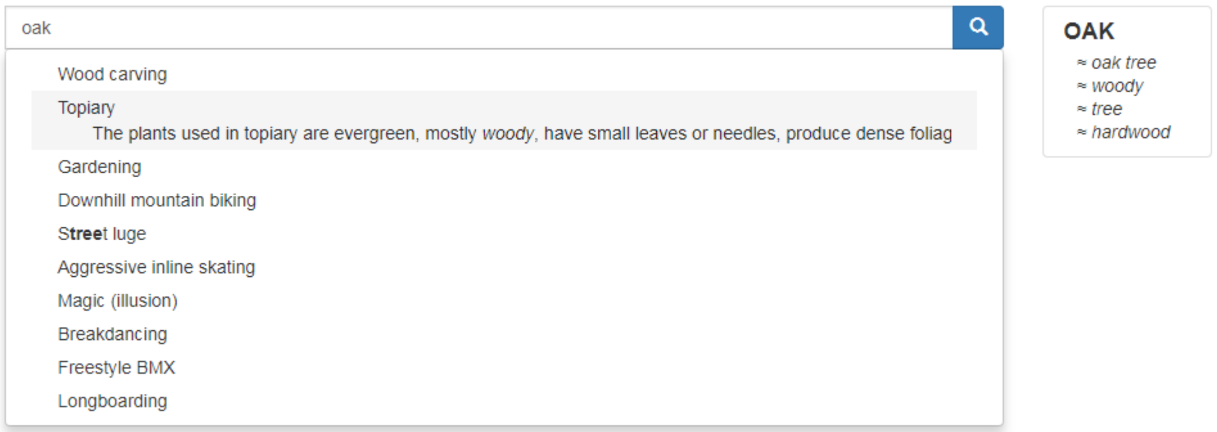
Dirtsurfing

Parkour

SWI
SWING

When a word is completed, the suggestions change from auto-complete to similar words. Sometimes, an entry only contains similar words. These are italicized instead of bolded.

(Btw, all those weird search results near the bottom? They have ‘street’ in them, which contains *tree*. This is not useful behavior for this search term, but may be useful for compound words like firetruck)


oak

Wood carving

Topiary
The plants used in topiary are evergreen, mostly *woody*, have small leaves or needles, produce dense foliag

Gardening

Downhill mountain biking

Street luge

Aggressive inline skating

Magic (illusion)

Breakdancing

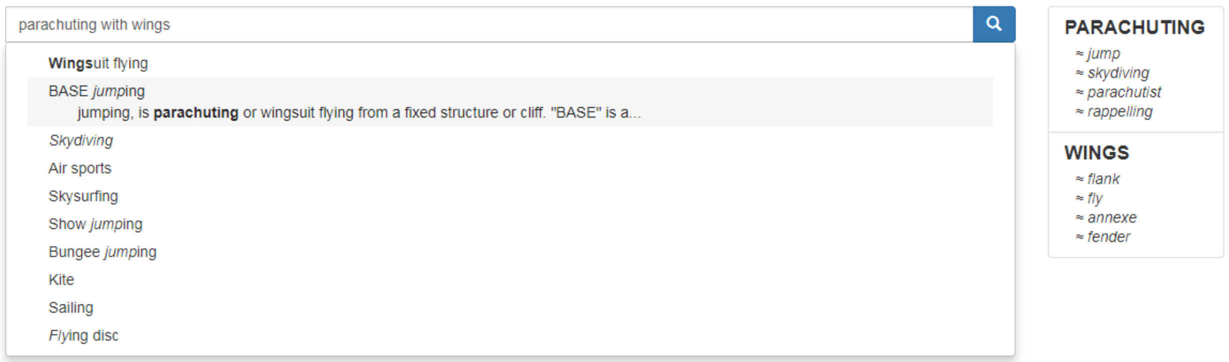
Freestyle BMX

Longboarding

OAK
≈ *oak tree*
≈ *woody*
≈ *tree*
≈ *hardwood*

The bar can handle multiple words and ignores less useful words. Here, 'with' is ignored. The highlighted word in the summary seems arbitrary, but it's actually the word with the highest "score" value.

A direct match, 'parachuting', is worth more than the similar word 'jumping'.


parachuting with wings

Wingsuit flying

BASE *jumping*

jumping, is **parachuting** or wingsuit flying from a fixed structure or cliff. "BASE" is a...

Skydiving

Air sports

Skysurfing

Show *jumping*

Bungee *jumping*

Kite

Sailing

Flying disc

PARACHUTING

≈ *jump*

≈ *skydiving*

≈ *parachutist*

≈ *rappelling*

WINGS

≈ *flank*

≈ *fly*

≈ *annexe*

≈ *fender*

How are similar and suggested words generated?

The similar and suggested words come directly from an API hosted by Datamuse.

Suggested: https://api.datamuse.com/sug?s=*word*

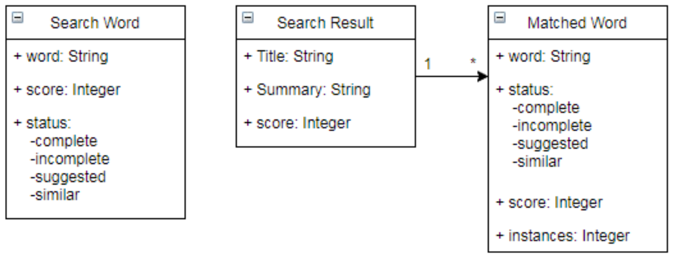
Similar: https://api.datamuse.com//words?ml=*word*

The algorithm

1. Grab the user's query.
2. Split it into words.
   a. Discard less useful words according to a list from http://www.stopwords.org
3. Categorize the words.
   a. Every word but the last is trusted as a "complete" word. (Unsafe, to be sure.)
   b. The last is either "incomplete" or "complete", as determined by the results from https://api.datamuse.com/sug?s=*word*
4. Generate related words.
   a. "similar" words are generated from sending "complete" words to https://api.datamuse.com//words?ml=*word*
   b. A "suggested" word is generated from sending "incomplete" words to https://api.datamuse.com/sug?s=*word*
5. Assign scores to the words, to simulate relevancy.
   a. "complete" words are worth the most.
   b. "similar" words are worth almost as much as complete words.
   c. "incomplete" words are not worth very much.
   d. "suggested" words are worth even less.
6. Each word found in a data entry adds that word's score to the entry's cumulative score according to a few rules:
   a. The first word found is worth more than the next words.
   b. If the first character matches, the entry receives a lot more points. (To reduce the 'tree is in street' problem.)
   c. Title matches are worth more than summary matches.
7. Sort the data entries by their score and display them to the user with matching words highlighted.

Data structures

Three main objects are used to hold information.



Difficulties

General Design Problems

Without good search results, user tests would likely not be very useful. So, in the desire to build a good search algorithm, the project quickly became mired in the technical approach to displaying well sorted entries by both the actual search query and similar search queries. Search is complex and it took a lot of time to reach a design that was useful at all.

The choice of data to search introduced another difficulty. The original intention of the similar queries approach was to be used on a set of frequently asked questions or help topics. This might allow many opportunities for a search query to be close but not quite match a topic. However, with the list of hobbies as the data set, opportunities for similar queries to be useful was limited, as each hobby's title was usually one word only. This is all to say that finding a hobby by name is more intuitive than finding it by related/similar words.

Showing the user what was happening with their search query was time-consuming to implement and never quite reached a crystal clear level of unobtrusive yet useful information.

The Datamuse API's similar words functionality is cool, but sometimes it returns strange, less than useful results. It's probably more useful if the data entries have keywords that contain similar words already. Then those can be a part of the search results. Auto-generating the keywords will be difficult, however.

Specific Functionality Issues

Because the algorithm will only search for similar words once it has deemed a word "complete" the results will tend to jump around.

There are some bugs where the summary doesn't get sliced correctly.

The API requests are synchronous, so the UI will have increased response times if either the number of requests becomes large (5+ "complete" words) or the Datamuse API is responding slowly.

Conclusion

The search works for the most part and I'm pretty happy with it. It shows you what it's searching for and highlights what it found. It was a fun and challenging project and while the original goals weren't met, I'm proud to have built a functional, if quirky, design.