# Write a HTTP service that provides an endpoint for fuzzy search / autocomplete of English words.

You are given a [dataset](https://drive.google.com/file/d/1dZdWIkjP6MNWFt-Umq8BfUU4AplpmRHX/view?usp=sharing) that contains 333,333 English words and the frequency of their usage in some corpus. A very small sample is shown below:

track   112385243

australia 112197265

discussion 111973466

archive 111971865

once 111882023

others 111397714

entertainment 111394818

agreement 111356320

format 111279626

Let us assume we’re building a web app where the user types in a **single** word from this list in a search box. We wish to autocomplete the input in the search box.

Your objective is to write a Python app using Django framework that exposes a single endpoint:

GET /search?word=<input>

where input is the (partial) word that the user has typed so far. For example, if the user is looking up procrastination, the service might receive this sequence of requests:

GET /search?word=pro

GET /search?word=procr

GET /search?word=procra

and so on.

The response should be a JSON array containing upto 25 results, ranked by some criteria (see below).

## Constraints

1. Matches can occur anywhere in the string, not just at the beginning. For example, eryx should match archaeopteryx (among others).
2. The ranking of results should satisfy the following:
   1. We assume that the user is typing the beginning of the word. Thus, matches at the start of a word should be ranked higher. For example, for the input pract, the result practical should be ranked higher than impractical.
   2. Common words (those with a higher usage count) should rank higher than rare words.
   3. Short words should rank higher than long words. For example, given the input environ, the result environment should rank higher than environmentalism.
      1. As a corollary to the above, an *exact match* should always be ranked as the first result.
3. The search algorithm you develop should ideally incorporate some form of a weighted average of all qualifying parameters. The perfect weights, in production systems, are however derived through the use of ML algorithms.

## Requirements

* Please add basic documentation for the project and use regular commits with proper commit messages.
* Please host the project in a publicly accessible location for testing like Heroku etc.

## Bonus

* Bonus marks depending on how fast your implemented endpoint works for our input test cases.
* Bonus marks for setting up a basic input box in the frontend which calls your API endpoint on the backend when some input is typed in it.

## Submission

Please submit the source code of your solution as a public GitHub repository.