Design and Integration of a Web Crawler for Query Processing

Maggie Barclay

Department of Computer Science, Illinois Institute of Technology

CS 429: Information Retrieval

Final Project Documentation

30 April 2022

Abstract

In this report, the project overview, design, architecture, operation, results, and test cases are presented. The objective of this project is to create a functional full-stack query processor, using data from a web scraper and information retrieval techniques to return relevant documents to a user.

The next steps for this project are to create an even more approachable user interface, and incorporate more complex techniques for search such as query suggestion (auto-fill).

I. <u>Overview</u>

A. Solution Outline

The proposed system is comprised of three main components:

1. A Scrapy based Crawler
    a) Will download web documents in html format
    b) The seed will be initialized using a seed URL/Domain
    c) A max number of pages / max depth will be set
2. An Indexer
    a) Will construct an inverted index in pickle format
    b) TF-IDF score / weight representation
    c) Cosine similarity
3. Flask based Processor
    a) Will handle free text queries in json format
    b) Query validation / error checking

c) Top-K ranked results

B. Relevant Literature

Relevant literature for this project includes context for the capabilities of these systems. Topics that require specific knowledge in information retrieval will be included, and topics that do not will not be focused on. This means relevant literature will include more about the algorithms behind the processing and indexing rather than web-crawling basics. This will include 1) Inverted Indexes, 2) TF-IDF, 3) Cosine Similarity, 4) Query Validation, and 5) Top-K Ranked Results

1) Inverted Indexes

An inverted index is a data structure that is critical in information retrieval. The general idea is this: A dictionary of terms is kept. For each term, a list is kept to track documents that the term appears in. Sometimes, the position in the document is also kept. Each of the items on this list is referred to as a posting (Manning 2018).

2) TF-IDF

TF-IDF is important to understand, as it is a method that helps to generate relevant results for a given query. It combines term frequency and inverse document frequency to produce a weight for each term in a document. Though the equations will be utilized in the code itself, it is important to mention in this documentation the reasoning behind why it is effective for this project. This score will be highest when a term occurs many times in a small number of documents, and lower when it occurs fewer times or in many documents. The documents can be viewed as vectors with components that each correspond to terms in the vocabulary (Manning 2018). These vectors enable comparisons which are the basis for retrieving only the most relevant results.

3) Cosine Similarity

   Cosine similarity is a standard way to check similarity between documents. The numerator takes the dot product, while the denominator represents that product of the documents' euclidean lengths (Manning 2018).

4) Query Validation

   A common query validation method is spelling correction (Manning 2018). This will be completed by utilizing k-gram indexes. More about the functionality and algorithm will be discussed in later sections.

5) Top-K Ranked Results

   Top-K results ranking fetches the ‘k” results with the highest matching score, indicating the highest relevance (Manning 2018).

C. Proposed System

The proposed system will consist of three parts: A web crawler, an indexer, and a processor. The design and architecture will be discussed in this report, however the general idea will be as follows: A web crawler gathers documents which the indexer will use to create an inverted index, and do further processing. This processing will help the processor to search the index and retrieve relevant documents based on a user given query.

II. Design

A. System capabilities

1. Web Crawler

The web crawler is constructed in the project_crawler folder, webby_spider.py. This ‘spider’ starts crawling at the domain 'https://nookipedia.com/wiki/List_of_villagers' to get a list of Animal Crossing villagers from the hit Nintendo Game, *Animal Crossing*. A depth is set by simply allowing the spider to crawl through the given number of villagers in the html table on the starting domain. This would be considered a breadth first search crawl of one level. The data of each villager is held in its own html file, in the htmlFiles folder.

The crawler saves local copies of the html files so that a local corpus is created. This allows the indexer to work separately from any web-crawling code, reducing the amount of times the sites are hit through web crawling in the development iterations.

2. **Indexer**

The indexer does several things. First, it parses the html files so that we can get the relevant information about our villagers. It constructs a “not inverted index” which is pickled for the processor to use (indexPickle). This is just some information we can spit out at the user once the actual inverted index helps us to know which villagers are most relevant to the query. TF-IDF is calculated, as well as a method for tokenizing queries. Several other pieces of information are pickled for use by the processor. These are invIndPickle (our inverted index of TF-IDF values), dfPickle (the document frequencies part of TF-IDF used for making the inverted index is called, not super necessary to pickle this but was going to use it in a pickled way earlier on in my implementation) and tokPickle (tokenized documents for use in making a k-gram index). Cosine similarity method is defined in the indexer as well.

3. **Processor**

The processor unpickles the previously mentioned pickles, then defines methods for spelling correction. This is done using a k-gram index, and then comparing k gram index. Bigrams are made from the query and the vocabulary terms to decide what the user actually meant. This is used when rendering the top k (top 10) most relevant villagers for the given query. If the user inputs ‘dog’, you should expect to see 10 dog villagers given back to the user on the web page. The html files are stored in the templates folder. Flask is used to create the application, so the process is retrieving input from the user, calculating the cosine similarity using the same method as in the indexer file, checking if spelling corrections are needed, and then outputting the correct information on the html file.

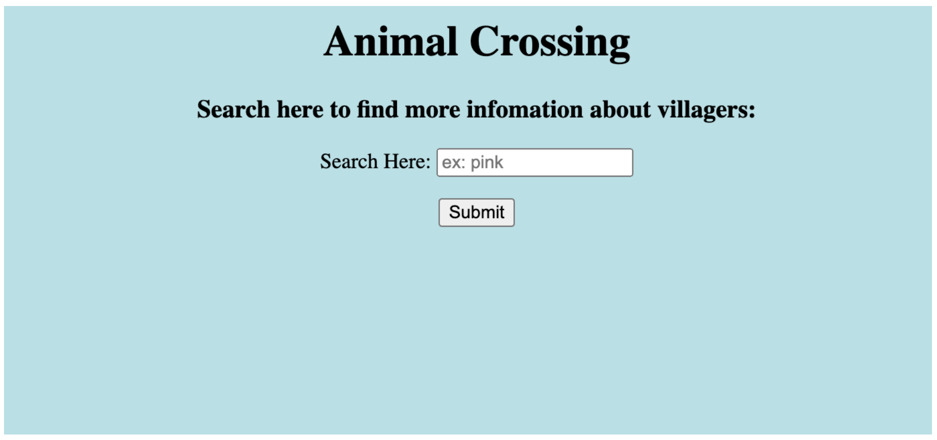
III. Architecture

A. Software Components & Interactions

1. Interaction between web crawler and indexer: no direct interaction, the html files saved locally by web crawler are used by the indexer
2. Interaction between web crawler and processor: none
3. Interaction between indexer and processor: the processor is dependent on the indexer, several of its methods are imported for use

B. Interfaces

A very simple interface for the user to enter a query. For example, if they enter pink they should expect villagers relating to this. See the test cases for example outputs.



IV. <u>Operation</u>

A. Software Commands

1. From the 'README' file:

- DOWNLOAD CODE LOCALLY
- INSTALL NECESSARY PACKAGES (use the generated requirements.txt to install)
- OPEN IN BROWSER http://127.0.0.1:5000/search
- YOU SHOULD NOT NEED TO RERUN THE WEB CRAWLER
    - IF DESIRED, RUN 'scrapy crawl webby' IN THE FIRST project_crawler DIRECTORY
- YOU SHOULD NOT NEED TO RERUN THE INDEXER
    - ANY PARTS THAT ARE UTILIZED BY THE PROCESSOR IN REALTIME SEARCH ARE ACCESSED AUTOMATICALLY THROUGH THE PROCESSOR.PY CODE
- IF DESIRED, RUN 'python indexerScript.py' (CS429Project > app > indexerScript.py)
- START FLASK APP BY RUNNING export FLASK_APP=processor AND THEN python -m flask run

V. <u>Conclusion</u>

A. Successes and Failures

1. **Successes**

    I completed the requirements of the project as given, successful searching and spelling corrections, visual representations to validate searches, and had a very fun time.

2. **Failures**

    The only thing in this project that I would consider a failure would be that I turned it in late. I am typing this right now past the due date, and will be pushing code past midnight. Though I'm happy with the application, I would have been happier if I managed my time more effectively and began the process earlier. I'm a "get it done in one go" type of person typically, but this project is a good example of why that doesn't always work (I had one weird bug that took me several hours to fix). Though I'm a little late, hopefully my very cute animal crossing villagers will persuade you that that's okay.

    I would have liked to do more extensive testing if I had more time (automated testing, coverage reports generated) instead of my very simple testing method of checking if things work myself because I'm running out of time. I would also have loved to do some of the extra credit components, but even before that I wish my web app didn't look like it was made in 2004.

    The only other thing I would say is that my code is in desperate need of some commenting. If I get the time between finals, I'll update in the repo. Hopefully though you'll be able to run seamlessly from my direction in this doc.

B. Outputs

1. Outputs of web scraper: html files in the htmlFiles folder
2. Outputs of indexer: pickle files, some command line outputs used for troubleshooting
3. Outputs of processor: rendered html webpage, some command line outputs used for troubleshooting

C. Caveats and Cautions

The only real issue I ran into during the process was with accessing the methods I needed from the indexer from the processor. Though it took me a while to solve, I made good use of pickling things, more than just the required inverted index pickling. This allowed me to grab information without recalculating everything from the indexer file. There's probably a better way to structure the code so that it's not necessary.

VI. Data Sources

A. Links and Downloads

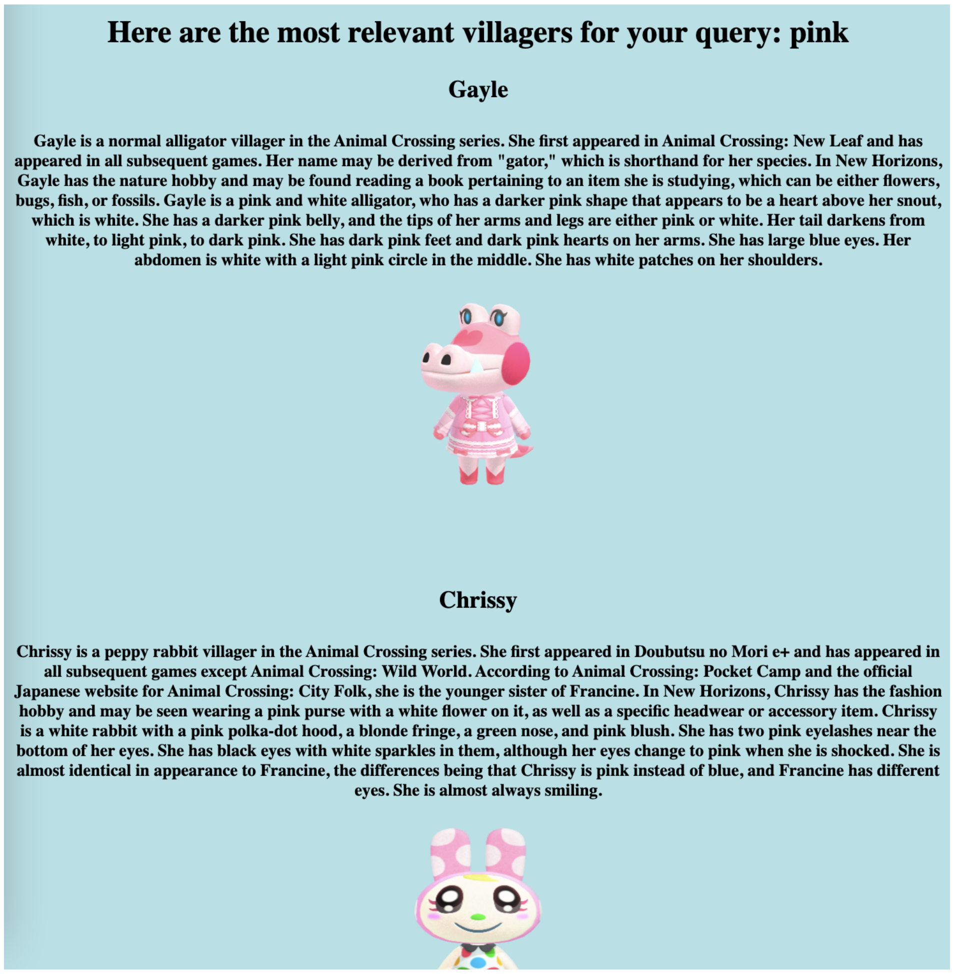
1. Links: https://nookipedia.com/wiki/List_of_villagers

2. Downloads: use the requirements.txt file to download the necessary packages

VII. Test Cases

1) Does the retrieved data make sense for a given query?

Yes, examples:



Here are the most relevant villagers for your query: pink

Gayle

Gayle is a normal alligator villager in the Animal Crossing series. She first appeared in Animal Crossing: New Leaf and has appeared in all subsequent games. Her name may be derived from "gator," which is shorthand for her species. In New Horizons, Gayle has the nature hobby and may be found reading a book pertaining to an item she is studying, which can be either flowers, bugs, fish, or fossils. Gayle is a pink and white alligator, who has a darker pink shape that appears to be a heart above her snout, which is white. She has a darker pink belly, and the tips of her arms and legs are either pink or white. Her tail darkens from white, to light pink, to dark pink. She has dark pink feet and dark pink hearts on her arms. She has large blue eyes. Her abdomen is white with a light pink circle in the middle. She has white patches on her shoulders.

Chrissy

Chrissy is a peppy rabbit villager in the Animal Crossing series. She first appeared in Doubutsu no Mori e+ and has appeared in all subsequent games except Animal Crossing: Wild World. According to Animal Crossing: Pocket Camp and the official Japanese website for Animal Crossing: City Folk, she is the younger sister of Francine. In New Horizons, Chrissy has the fashion hobby and may be seen wearing a pink purse with a white flower on it, as well as a specific headwear or accessory item. Chrissy is a white rabbit with a pink polka-dot hood, a blonde fringe, a green nose, and pink blush. She has two pink eyelashes near the bottom of her eyes. She has black eyes with white sparkles in them, although her eyes change to pink when she is shocked. She is almost identical in appearance to Francine, the differences being that Chrissy is pink instead of blue, and Francine has different eyes. She is almost always smiling.

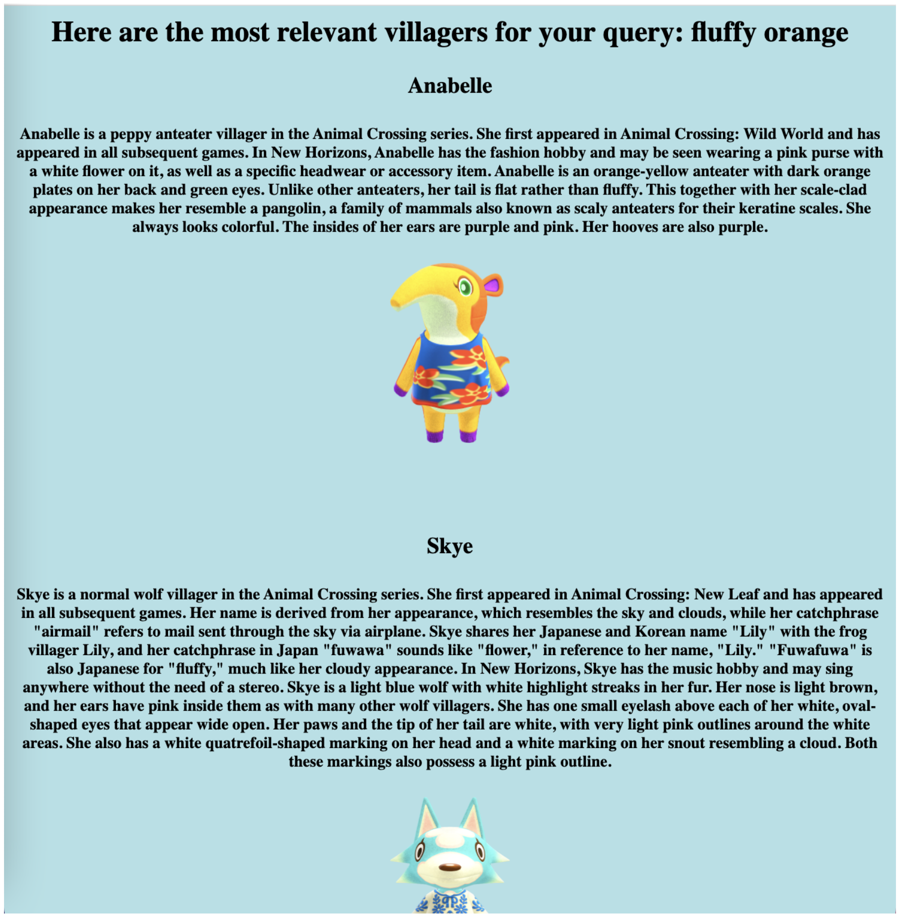
Here are the most relevant villagers for your query: bunny

Bunnie

Bunnie is a peppy rabbit villager in the Animal Crossing series who appears in all games to date. Her name is derived from bunny, which relates to her species. In New Horizons, Bunnie has the fashion hobby and may be seen wearing a pink purse with a white flower on it, as well as a specific headwear or accessory item. Bunnie is a chalky, burnt-orange rabbit with a creamy beige face that makes her resemble apple rabbits, with the eyes looking like apple seeds, and arms that are tipped the same color as her head; her bunny tail and legs are the same color, too. Her face has pink blush and a big smile. Her eyes are slightly anime-styled in a pointy diamond-like way with a sparkle in them. Her ears are peachy orange on the inside and held high in the air.

Snake

Snake is a jock rabbit villager in the Animal Crossing series who appears in all games to date. His name may be derived from sneak, an anagram of snake, which relates to his ninja-like appearance. His catchphrase, "bunyip," is a play on bunny, and is also the name of a monster sometimes depicted with serpent-like characteristics in Australian Aboriginal mythology. In New Horizons, Snake has the fitness hobby and will wear Sporty Shades and lift heavy dumbbells while exercising. Snake is a pink rabbit who wears a hood that gives him the appearance of a ninja. He has purple legs and arms, large yellow cheeks, and white on the insides of his ears.



Here are the most relevant villagers for your query: fluffy orange

Anabelle

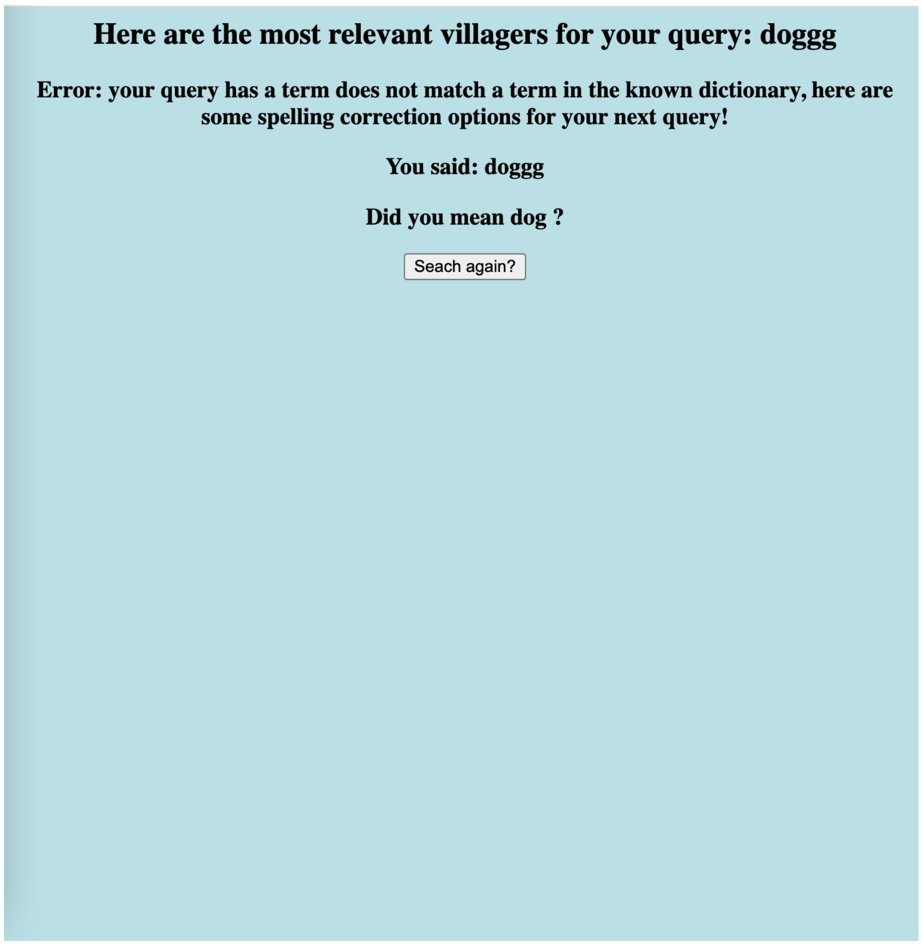
Anabelle is a peppy anteater villager in the Animal Crossing series. She first appeared in Animal Crossing: Wild World and has appeared in all subsequent games. In New Horizons, Anabelle has the fashion hobby and may be seen wearing a pink purse with a white flower on it, as well as a specific headwear or accessory item. Anabelle is an orange-yellow anteater with dark orange plates on her back and green eyes. Unlike other anteaters, her tail is flat rather than fluffy. This together with her scale-clad appearance makes her resemble a pangolin, a family of mammals also known as scaly anteaters for their keratine scales. She always looks colorful. The insides of her ears are purple and pink. Her hooves are also purple.

Skye

Skye is a normal wolf villager in the Animal Crossing series. She first appeared in Animal Crossing: New Leaf and has appeared in all subsequent games. Her name is derived from her appearance, which resembles the sky and clouds, while her catchphrase "airmail" refers to mail sent through the sky via airplane. Skye shares her Japanese and Korean name "Lily" with the frog villager Lily, and her catchphrase in Japan "fuwawa" sounds like "flower," in reference to her name, "Lily." "Fuwafuwa" is also Japanese for "fluffy," much like her cloudy appearance. In New Horizons, Skye has the music hobby and may sing anywhere without the need of a stereo. Skye is a light blue wolf with white highlight streaks in her fur. Her nose is light brown, and her ears have pink inside them as with many other wolf villagers. She has one small eyelash above each of her white, oval-shaped eyes that appear wide open. Her paws and the tip of her tail are white, with very light pink outlines around the white areas. She also has a white quatrefoil-shaped marking on her head and a white marking on her snout resembling a cloud. Both these markings also possess a light pink outline.

2) Does the spelling correction work?

Yes



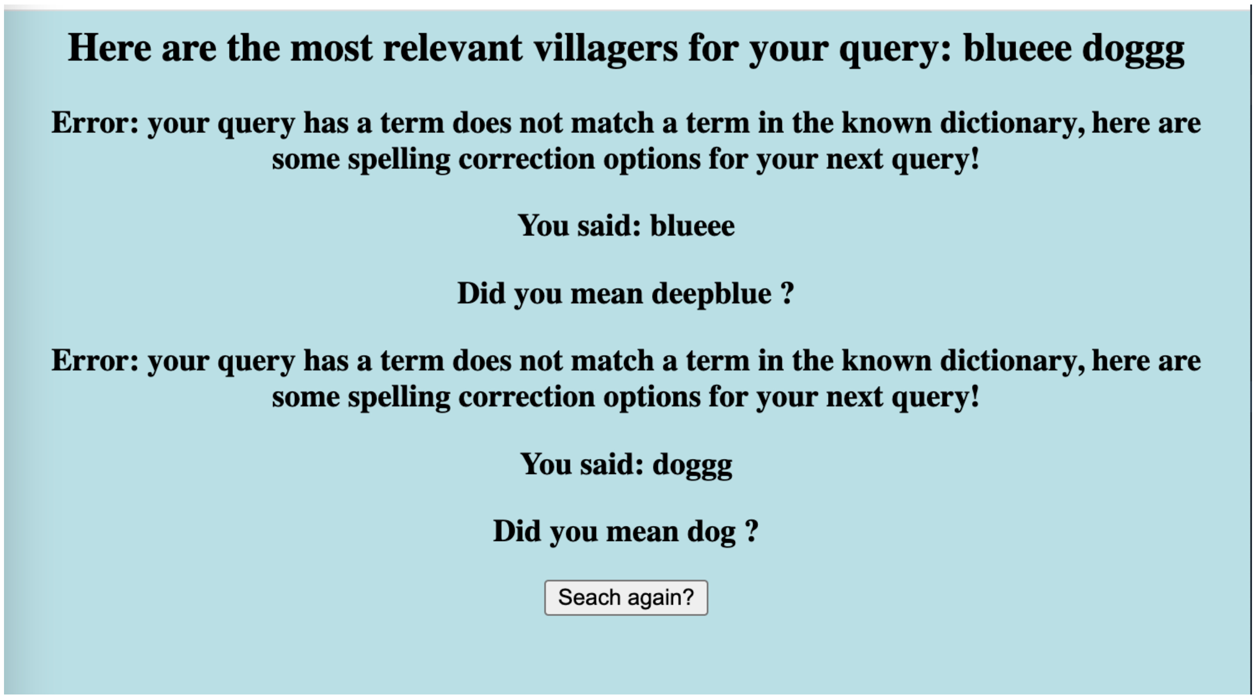
Here are the most relevant villagers for your query: doggg

Error: your query has a term does not match a term in the known dictionary, here are some spelling correction options for your next query!

You said: doggg

Did you mean dog ?

3) Does the spelling correction work for multiple terms?


Here are the most relevant villagers for your query: blueee doggg

Error: your query has a term does not match a term in the known dictionary, here are some spelling correction options for your next query!

You said: blueee

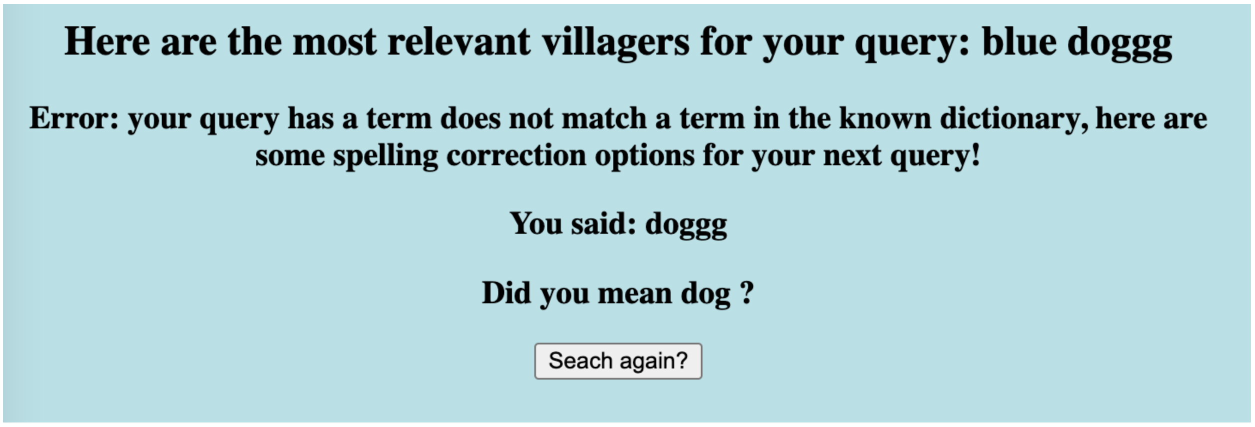
Did you mean deepblue ?

Error: your query has a term does not match a term in the known dictionary, here are some spelling correction options for your next query!

You said: doggg

Did you mean dog ?

4) Does the spelling correction work for multiple terms where some are and some are not spelled correctly?


Here are the most relevant villagers for your query: blue doggg

Error: your query has a term does not match a term in the known dictionary, here are some spelling correction options for your next query!

You said: doggg

Did you mean dog ?

5) Does the button to search again work?

Yes

VIII. Source Code

A. **My code: https://github.com/maggiebarclay/CS429Project**

B. **Dependencies**

1. **Flask: https://flask.palletsprojects.com/en/2.1.x/**
2. **Scrapy: https://docs.scrapy.org/en/latest/**
3. **Pickle: https://docs.python.org/3/library/pickle.html**
4. **NLTK: https://www.nltk.org/**
5. **BeautifulSoup: https://beautiful-soup-4.readthedocs.io/en/latest/**
6. **OS: https://docs.python.org/3/library/os.html**
7. **Numpy: https://numpy.org/**

IX. References

Manning, Christopher, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2018.