GeoTravel: Folium-based Search Engine

CS483 - Instructor: Ben McCamish - Web Data

Rebollar, Emmanuel
WSU Student
Vancouver, WA
emmanuel.rebollar@wsu.edu

Ramirez, Megan
WSU Student
Vancouver, WA
megan.ramirez@wsu.edu

Serea, Samuel
WSU Student
Vancouver, WA
samuel.serea@gmail.com

Cotton, Ronald
WSU Student
Vancouver, WA
ronald.cotton@wsu.edu

ABSTRACT

GeoTravel is a minimalist search engine implemented in Python using various libraries like Whoosh for data indexing, Flask for database interfacing, and Jinja for dynamic web page building. The search engine displays results and information on lakes, mountains, national parks, and waterfalls as per the user's query. The data collected in the backend database is scraped from Wikipedia's API and parsed into four data tables built in SQLite, each containing information about their respective topics.

General Terms

Web Data

Keywords

ACM proceedings, LaTeX, python, whoosh, jinja, flask, folium.

1. INTRODUCTION

GeoTravel is a website that allows users to search for nature destinations all around the world. The database contains information on lakes, mountains, national parks, and waterfalls. All of the data has been scraped from Wikipedia and saved in a SQLite database. An index is then created over the database using Whoosh. When the user enters a query, via the web interface, the index is searched and returns the most relevant results. These results are then displayed on the web interface, using both a map and an image table. The map display features a marker for each of the individual results. The image table shows a picture of each result which, when clicked on, will popup more information about the location. Each image also has a button that will center the map on that location as well as a button that will take the user to the Wikipedia page that the information was originally sourced from. GeoTravel was made to give users an easy way to visualize and interact with the results of their query.

2. FEATURES

2.1 Hyperlink

A link for each tuple's Wikipedia page in the database has been collected using Wikipedia's API. When the results of a user query are returned, an image table is displayed on the right side of the screen. Each image has a button in the top right labeled "WIKIPEDIA". Clicking this button will pass the Wikipedia URL to a function to check if the link is expired. If the link is expired a plain web page will be brought up saying, "Wikipedia page not found".

Figure 1: Example of an image block in the image table.

2.2 Images

An image for each tuple in the database has been collected, if one is available, using Wikipedia's API. If the tuple does not have an available image, a picture of a globe is added as a default. After the user searches the database, a table on the right side of the page is returned with an image and title of the location.

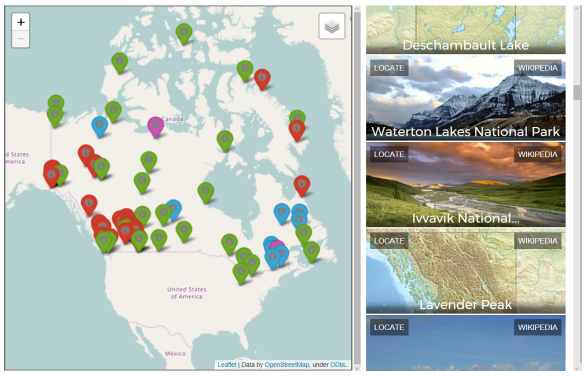

Figure 2: Example of the image table on the right side of website.

Hovering over an image with the mouse turns it slightly gray indicating to the user they can interact with the image. If the user clicks on the image, a screen pops up giving the user more information and showing them the full image.

2.3 Interactive map

An interactive map displaying the locations of each result is featured on GeoTravel. The map is displayed on the left half the web page and displays all of the produced locations. The map displays a marker indicating the location of each of the results. When a marker is clicked, the name of the tag is displayed above. The markers are color coded as follows:

- Blue - Lakes
- Red - Mountains
- Green - National Parks
- Purple - Waterfalls

Each image on the right side of the page has a button in the top left labeled "LOCATE". Clicking this button will center the map on the location associated with that image.

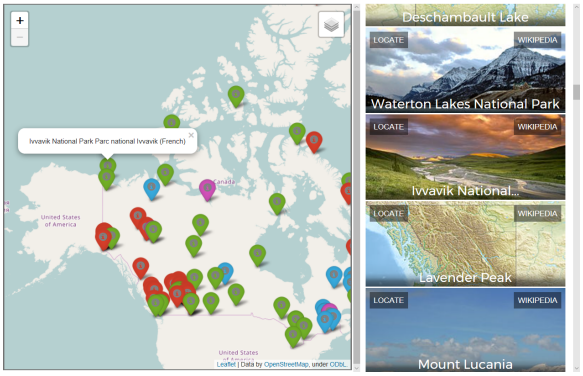

Figure 3: Example of the marker popup and all the different colors of markers on map.

3. SEARCH ENGINE

The searching feature in **GeoTravel** uses a dictionary built over elements in the title, location, description, and category attributes of the database. A tokenizer used in the dictionary and user input is created through filtering and analyzing modules that converts all text to lowercase, removes special characters like accents, uses an English stop-filter, and uses a stem-filter.

3.1 Searching

The user can enter a search into the search bar at the top of the HTML web page. There are also four check boxes, lake, mountains, national parks and waterfalls, that allow the user to narrow their search. Once the user hits enter the map is loaded on the left hand side of the page and on the right hand side of the page a table of results is returned. The user can scroll through all the entries in the table without losing sight of the map. After the results are returned, the user's query is still entered in the search bar. For example, a user selected national parks and typed "Canada" in the search bar and hits enter. After the results are returned, only the national parks check box is checked and "Canada" is still present in the search bar.

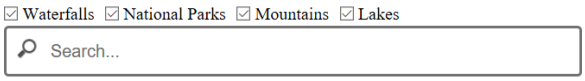

Figure 4: Example of the search bar and check boxes.

3.2 Database

The web scraper uses the Wikipedia API to collect information on lakes, mountains, national parks, and waterfalls, storing results into a SQLite database. The database is 5.52MB and contains a total of 6042 tuples and is broken up as follows:

- 2842 - Lakes
- 1275 - Mountains
- 1333 - National Parks
- 592 - Waterfalls

4. RELATED WORK

4.1 Degree Conversion

The following formula is used for degree-decimal conversion[6] for longitude and latitude from minutes and seconds:

$$\text{dec. degrees} = (\ \text{deg} + \text{min}/60 + \text{sec}/3600\ )$$
$$\textit{dec. degrees times -1 when S or W}$$

The result is returned with 6 decimal degrees of precision[5].

4.2 Whoosh

Whoosh[1] is a fast, full-text indexing, and searching library implemented in pure Python. Programmers can use it to easily add search functionality to their applications and websites.

Some of Whoosh's features include:

- Python APIs
- Pure-Python; No compilation or binary packages needed
- Fielded indexing and search
- Fast indexing and retrieval — faster than any other pure-Python, scoring, full-text search solution[2]
- Text analysis, storage, posting format, etc.

4.3 Okapi BM25

Okapi BM25[7] is a retrieval function which ranks documents by their relevance to a given search query. Based on a **TD-IDF** *(Term Frequency, Inverse Document Frequency)*, BM25 features include *nonlinear term-frequency saturation* and *field-length normalization*. Inverse Document Frequency distinguishes between lower valued common words versus high value uncommon words. Term Frequency recognizes the frequency of a search term in a document. Nonlinear term-frequency saturation prevents keywords receiving artificially high rankings when the word is used multiple times. By default, Whoosh employs BM25 in it's `whoosh.scoring` module and may be fine-tuned for performance.

4.4 HTML

GeoTravel's frontend involves two HTML5 pages interlaced with Jinja2's template engine. Flask serves both web pages as queries and results are handled by the Whoosh search engine. The landing page, `index.html`, is composed of a scalable backdrop with a simplistic search query box rendered in the center of the screen. Results are then processed by flask and the results are tabulated onto the results page which is found in `welcome_page.html`.

Both pages have favicons generated by realfavicongenerator.net which creates optimized favicons for all browsers and operating systems. The results page is broken into three distinct sections. The largest section is an iframe containing a javascript map generated by Folium. On the right is an images table, representing the search results. At the top, a search box and four checkboxes are displayed.

4.4.1 IFrame (map.html)

The iframe was bound to the LeftContent div containing `map.html`. The iframe references this page, as folium places markers on the search query results generated by Whoosh. The markers are colored based on the type of tuple. To fix an issue we had with a vertical scrollbar when rendering the iframe, the CSS style was shimmed with the styling of a width of 99% and a height of 99%.

4.4.2 Landing Page (index.html)

The landing page chooses a random background image to be loaded using these Jinja statements:

```
{% set bgnum = range(1,8) | random | string %}
{% set bgfn = 'images/background' + bgnum + '.jpg' %}
```

Input checkboxes are hidden from the user which allows jinja to process the page on the results page without the original values present.

```
<input type="hidden" ... >
```

4.4.3 Results Page (welcome_page.html)

The results page checkboxes are selected or unselected based on which checkboxes were selected or deselected previously. When results are returned by flask, jinja iterates through the tuples of titles, descriptions, images, links, longitudes, and latitudes and generates each image, all the links associated with each search result, and content for the modal. The modal is generated with sweetAlert, a simple javascript package which enables us to render a centered message which allows content to be in HTML. If no queries are returned, Jinja instead renders everything after the % else % command:

```
{% for titles, descriptions, images, links, lon, lat
   in results %}
... each tuple result
{% else %}
...No Results
{% endfor %}
```

4.4.4 Images

The background images used in the landing page were high-resolution public domain NASA photos. The image size was optimized by using mozjpeg cjpeg, rendered at a quality of 80% and all metadata stripped. This is the command we used to accomplish the task:

```
cjpeg -quality 80 -copy none <infile> <outfile>
```

Images were placed in the center of the background with a fixed position using CSS.

4.5 Folium

Folium delegates as an intermediary for python and Leaflet.js and generates an interactive map. This map is embedded into the **GeoTravel** website with the original coordinates converted from Degree Minutes Seconds *Degree-Minutes-Seconds* format to *Degree-Decimal* format. Parameters are passed to Folium's tileset function producing colored markers. Colors are designated locations and are switchable by checkboxes.

4.6 Flask

Flask[3] is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. It is BSD licensed.

Flask is called a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies, and several common framework related tools.

4.7 Jinja

Jinja[4] is a template engine for the Python programming language. It is similar to the Django template engine but provides Python-like expressions while ensuring that the templates are evaluated in a sandbox. It is a text-based template language and thus can be used to generate any markup as well as source code.

The Jinja template engine allows customization of tags, filters, tests, and global variables. Also, unlike the Django template engine, Jinja allows the template designer to call functions with arguments on objects. Jinja is Flask's default template engine.

4.8 SQLite

SQLite is a relational database management system contained in a C programming library. In contrast to many other database management system, SQLite is not a client-server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages.

5. CONCLUSION AND FUTURE WORK

The **GeoTravel** search engine behaves like any other search engine. The user types keywords into the search bar and is presented with results relating to their search. The data searchable by the user consist of four main categories, lake, mountains, national parks, and waterfalls, which are scraped from Wikipedia and stored in a SQLite database. On the front end, a HTML web page is built by Flask and Jinja displaying the results on a web page. The web page presents the results in both a map and image table. The map depicts the specific locations of the results by showing a marker over the location of each result. Clicking on a marker displays the name of the location for user readability. The image table shows an image of each location and allows the user to visit the Wikipedia page, locate the place on the map, or view more information by clicking on the image.

In the future, we may look into making the search engine faster as well as adding timezone information and weather details.

6. REFERENCES

[1] M. Chaput. Whoosh 2.7.4 documentation, 2016.
[2] M. Chaput. Whoosh 2.7.4 homepage, 2016.
[3] A. Ronacher. Flask documentation, 2016.
[4] A. Ronacher. Jinja2 documentation, 2017.
[5] Wikimedia. Wikipedia: Decimal degrees, 2017.
[6] Wikimedia. Wikipedia: Geographic coordinate conversion, 2017.
[7] Q. S. Zhendong Shi, Jacky Keung. An empirical study of bm25 and bm25f based feature location techniques. *InnoSWDev 2014*, 322(10):106–114, 2014.