README.md

# Quote Finder

## Authors

- Team Name | **Lunar-Tsai**
  - Manuel Suarez Lunar | manuel6@illinois.edu
  - Wei-Lun (Will) Tsai | wltsai2@illinois.edu <-- team captain
- University of Illinois Urbana-Champaign | Fall 2023
- CS 410: Text Information Systems | ChengXiang Zhai
- Code Repo | github.com/wlltsai/quote-finder
- Demo Video | drive.google.com/file/d/1bw_fsboUJzrEsjHlbS7dBuFE5-w_5QDTT/view?usp=sharing
- PPT slides | github.com/wlltsai/quote-finder/blob/main/course-deliverables/project-final-presentation-slides.pptx

## Overview

Quote Finder is a unique search engine / text retrieval tool specifically tailored for literature and quote enthusiasts. Users input a particular sentiment or emotion and then Quote Finder searches the catalog of famous quotes available on goodreads.com/quotes to return a ranked list of quotes and respective authors resonating with the user's specified sentiment.

## Self-Evaluation

We have completed four of our initially proposed goals with the implementation of a working web crawler, preprocessor, searcher, and web application. These tasks made up a bit more than the 20 hours of coding per team member that we had originally estimated due to cycles being spent on integrating the searcher with the web app and then debugging. Thus, we weren't able to complete our two stretch goals of implementing a feedback component as well as hosting the application on Azure. However, we were able to complete the core functionality of our application with a satisfactory set of initial results. In addition to applying the methods learned in this course, we also got to learn how to build a web crawler, leverage a sentiment library, and build a web application. Aside from our two stretch goals, the only other enhancements we would make given more time and resources are to extend sentiment analysis to the quotes themselves as well as tweaking the ranker to get further improvement in the quality of the results.

## Architecture

The architecture consists of three main backend components: (1) web crawler, (2) preprocessor, (3) searcher, (4) web application, and (5) user interface. The web crawler collects raw quotes data from the Goodreads quotes website, which is then processed by the preprocessor to extract, normalize, and tokenize the quotes along with their corresponding metadata. The searcher parses the user sentiment input, builds an inverted index, then searches against the index to return a set of ranked quotes based on relevance to a given query. A web application (API) is built on top of these components, which is then fronted by a user interface for users to interact with the application.

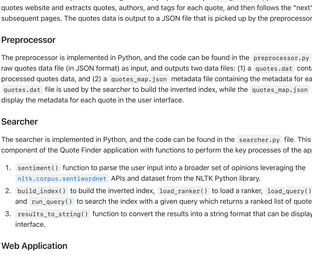



*Figure 1: Overall architecture diagram for Quote Finder*

### Distribution of Work

Between the two team members, the coding and documentation work was distributed as follows:

- Manuel Suarez Lunar
  - Web crawler
  - Web application
  - User interface
  - Powerpoint presentation
- Wei-Lun (Will) Tsai
  - Preprocessor
  - Searcher
  - Code documentation

### Web Crawler

The web crawler is implemented using the Scrapy Python library in `web_crawler.py`. It crawls the Goodreads quotes website and extracts quotes, authors, and tags for each quote, and then follows the "next" link to crawl subsequent pages. The quotes data is output to a JSON file that is picked up by the preprocessor.

### Preprocessor

The preprocessor is implemented in Python, and the code can be found in the `preprocessor.py` file. It takes a raw quotes data file (in JSON format) as input, and outputs two data files: (1) a `quotes.dat` containing the processed quotes data, and (2) a `quotes_map.json` metadata file containing the metadata for each quote. The `quotes.dat` file is used by the searcher to build the inverted index, while the `quotes_map.json` file is used to display the metadata for each quote in the user interface.

### Searcher

The searcher is implemented in Python, and the code can be found in the `searcher.py` file. This is the core component of the Quote Finder application with functions to perform the key processes of the application:

1. `sentiment()` function to parse the user input into a broader set of opinions leveraging the `nltk.corpus.sentiwordnet` APIs and dataset from the NLTK Python library.
2. `build_index()` to build the inverted index, `load_ranker()` to load a ranker, `load_query()` to load a query, and `run_query()` to search the index with a given query which returns a ranked list of quotes as the result.
3. `results_to_string()` function to convert the results into a string format that can be displayed in the user interface.

### Web Application

The web application is implemented using the Flask Python library in `webapp.py`. It exposes a set of API endpoints that are called by the user interface to invoke the searcher, and returns the results to the user interface.

### User Interface

The user interface is implemented using HTML templates in the `templates` directory. It consists of an `index.html` home page with a search bar for the user to input a sentiment, and an `output.html` page that displays the ranked list of quotes returned by the searcher. These UI pages are rendered by the `webapp.py` Flask web application.

## Setup Instructions

1. Clone the code repository and navigate to the project directory:

   ```
   git clone https://github.com/wlltsai/quote-finder.git
   cd quote-finder
   ```

2. Install Conda and Git if you don't already have them installed.

   > Note: if you are running this on an Apple Silicon Mac (e.g. M1, M2 chips), adjust your Conda config after installation:
   >
   > ```
   > conda config --env --set subdir osx-64
   > ```
   >
   > Note: if you are running this on a Linux machine, adjust your Conda environment variables after installation:
   >
   > ```
   > export CONDA_DIR=conda info | grep -i 'base environment'
   > source $CONDA_DIR/etc/profile.d/conda.sh
   > ```

3. Create and activate a Python 3.5 Conda environment:

   ```
   conda create -n py35 python=3.5
   conda activate py35
   ```

4. Create and activate a Flask environment for the web application:

   ```
   python -m venv .venv
   . .venv/bin/activate
   ```

5. Install the project dependencies:

   ```
   pip install -r requirements.txt
   ```

6. Run the web crawler and preprocessor to prepare the quotes data:

   ```
   python web_crawler.py
   python preprocessor.py
   ```

7. Run the web app:

   ```
   python webapp.py
   ```

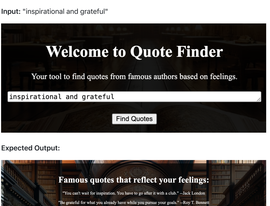
8. Navigate to http://127.0.0.1:5000/ in a web browser to interact with the Quote Finder web application.

## Example Test Cases

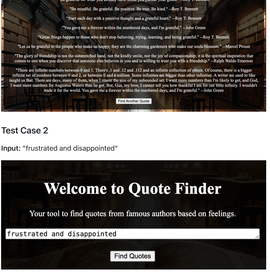
Here are some example queries to try out in the Quote Finder web application:

### Test Case 1

**Input:** "inspirational and grateful"

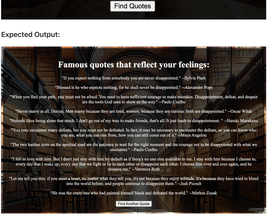



**Expected Output:**




### Test Case 2

**Input:** "frustrated and disappointed"




**Expected Output:**