1) An overview of the function of the code (i.e., what it does and what it can be used for).

The code provides the information below when a user enter a search term in the Chrome extension:

1. BM25 scores for first 30 documents in Google search and re-rank in descending order;
2. Top 5 most frequently used words and the corresponding word frequency for each document;
3. 5 topics covered in the 30 documents, and each topic is represented by the first 10 words.

It can be used for users wanting to modify their search query and wanting to find results that have to do mainly with the topic that satisfies their needs. It can also give a user an overview of each document as well.

2) Documentation of how the software is implemented with sufficient detail so that others can have a basic understanding of your code for future extension or any further improvement.

There are 2 components covered in this project: Python backend, and Chrome extension front end.

In the Python backend, we use [Django](https://www.djangoproject.com/) as the web framework and the tool to create a local host for the Chrome extension to run. In the ```mysite``` subfolder, there are multiple files created by the Django command line following the Django project file structure. There are 3 pivotal files implementing the actual backend functionalities that we should keep an eye on.

****mysite/intelligentgroup/document.py****
This file is used for creating a document object which has several attributes and functions. The attributes include the URL, raw content, list of words, word frequency of a document. The functions include scraping the web document, creating a word list, applying preprocessing steps such as lemmatization and stop word filtering. As well as calculating the word frequency.

mysite/intelligentgroup/search_action.py
The search action file is used for creating a search action object. A search action is defined by a query term and number of documents retrieved. The attributes include the search query term, number of web pages, and corpus. The functions include building the corpus, calculating the bm25 scores for all the documents and performing LDA topic modeling.

mysite/intelligentgroup/view.py
This file creates the view of the Web page, it calls objects & functions in ```search_action.py``` and ```document.py``` to perform the backend tasks, including BM25 scores calculation and re-ranking, word frequency calculation and topic modeling, then wrap up the result with HTML language to show in the web page.
As an experimental project, the webpage will be rendered in a local host. The URL of the page follows the format of:
http://127.0.0.1:8000/intelligentgroup/search_action_analysis/?query=<query>.

In the Chrome Extension front end, the files are all included in the ```chrome_extension``` subfolder.

mysite/chrome_extension/manifest.json

The extension's manifest is the only required file that must have a specific file name: manifest.json . It also has to be located in the extension's root directory. The manifest records important metadata, defines resources, declares permissions, and identifies which files to run in the background and on the page.

The JSON file describes the important configuration for the extension.

mysite/chrome_extension/popup.html
This file is the front end UI determined by HTML. It includes a text box to input a query term, and a submit button.

mysite/chrome_extension/popup.js
When a click action is processed from the extension UI, it sends the query which the user input to the background to get the full URL, and then opens the URL in a new Chrome tab. To make sure the web page to be successfully rendered, you will need to make sure the local host is active by running the command ```python manage.py runserver```. Detailed instructions will be provided later.

mysite/chrome_extension/backend.js
The file is constructed by Chrome API with a listener, it waits for a request to be sent and then sends the full URL as the response back to the popup.

3) Documentation of the usage of the software including either documentation of usages of APIs or detailed instructions on how to install and run a software, whichever is applicable.

How to install dependencies and set up local host:

1. Clone the Github Repo:

```
git clone https://github.com/lanceyjt/CourseProject_IntelligentGroup.git
```

2. Set up & activate Conda environment (Replace <env> by a name of your choice):

<u>If you are using MAC</u>

```
conda config --add channels conda-forge
conda create -n <env> --file requirements_conda.txt
conda activate <env>
```

<u>If you are using Windows</u>

```
conda config --add channels conda-forge
conda install django
conda install html2text
conda install scikit-learn
conda install requests
conda install nltk
conda install bs4
conda create -n <env> python=3.9
conda activate <env>
```

3. Install packages only viable through PIP:

```
pip install google
pip install rank_bm25
```

4. Run Django Server:

```
cd mysite
python manage.py runserver
```

By running the command above, you should be able to see a message like “Starting development server at http://127.0.0.1:8000/”.

To experience the backend functionality, you can input the URL:http://127.0.0.1:8000/intelligentgroup/search_action_analysis/?query=<query>.

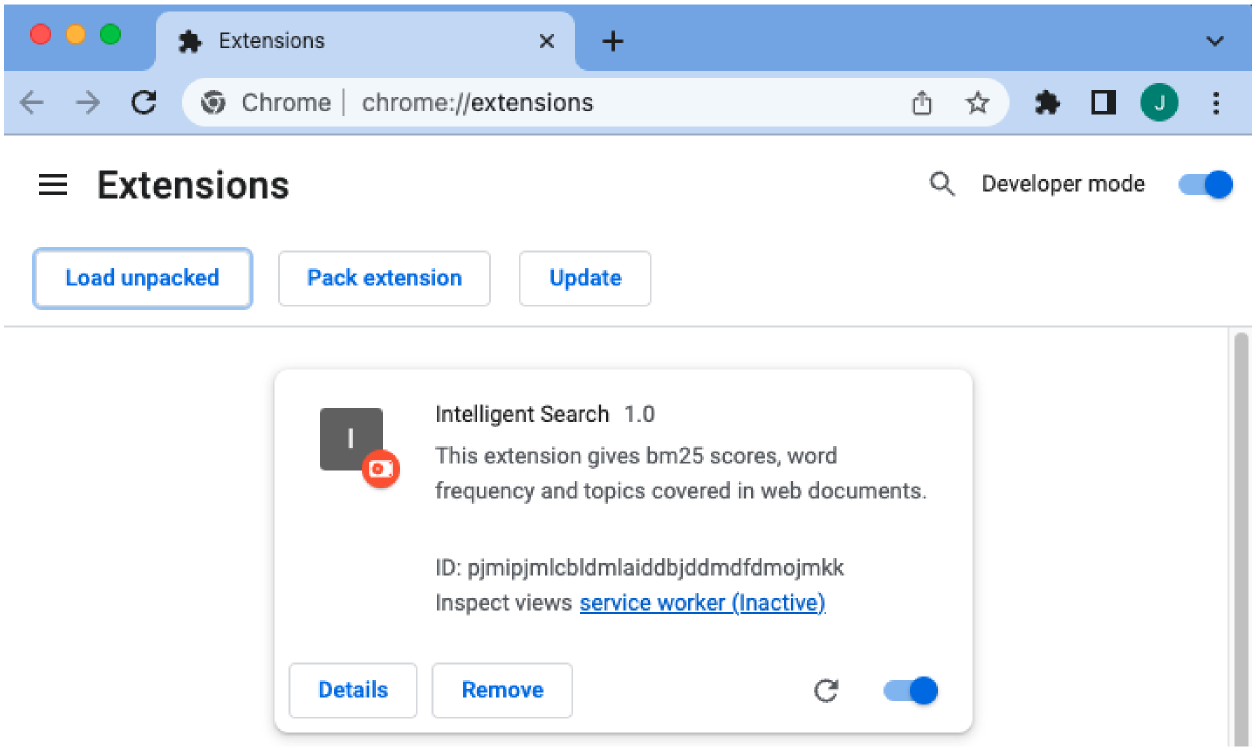
You can replace the query at the end with a search term query you would like to analyze(ex. http://127.0.0.1:8000/intelligentgroup/search_action_analysis/?query=bm25).

Please give this a few seconds of your time as it is processing your search query in the backend and will output the bm25 score, word frequency, and topics.

How to install and run application via chrome extension:

After you have successfully done the steps in the last section, please follow the steps below to run the Chrome Extension.

1. Click the three dots token at the top right corner of the Chrome browser.
2. Go to “More Tools” → “Extensions”
3. Toggle “Developer mode” at the top right corner if it isn’t already activated
4. Click “Load Unpacked” at the top left corner
5. A file explorer window will pop-up
6. Go to the “chrome_extension” subfolder from the code cloned from git, and click “Select”
7. You should be able see an extension named as “Intelligent Search 1.0” now



Intelligent Search 1.0

This extension gives bm25 scores, word frequency and topics covered in web documents.

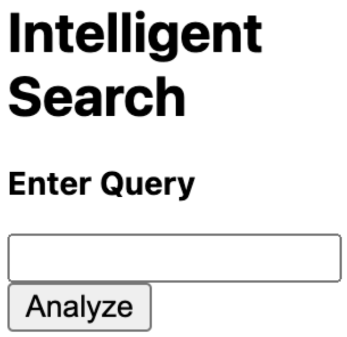
ID: pjmipjmlcbldmlaiddbjddmdfdmojmkk

Inspect views service worker (Inactive)

8. Make sure the extension is turned on by toggling the button at the bottom right

Important Note: To make sure the Chrome Extension works properly, please make sure the local host created by Django in the previous step is active.

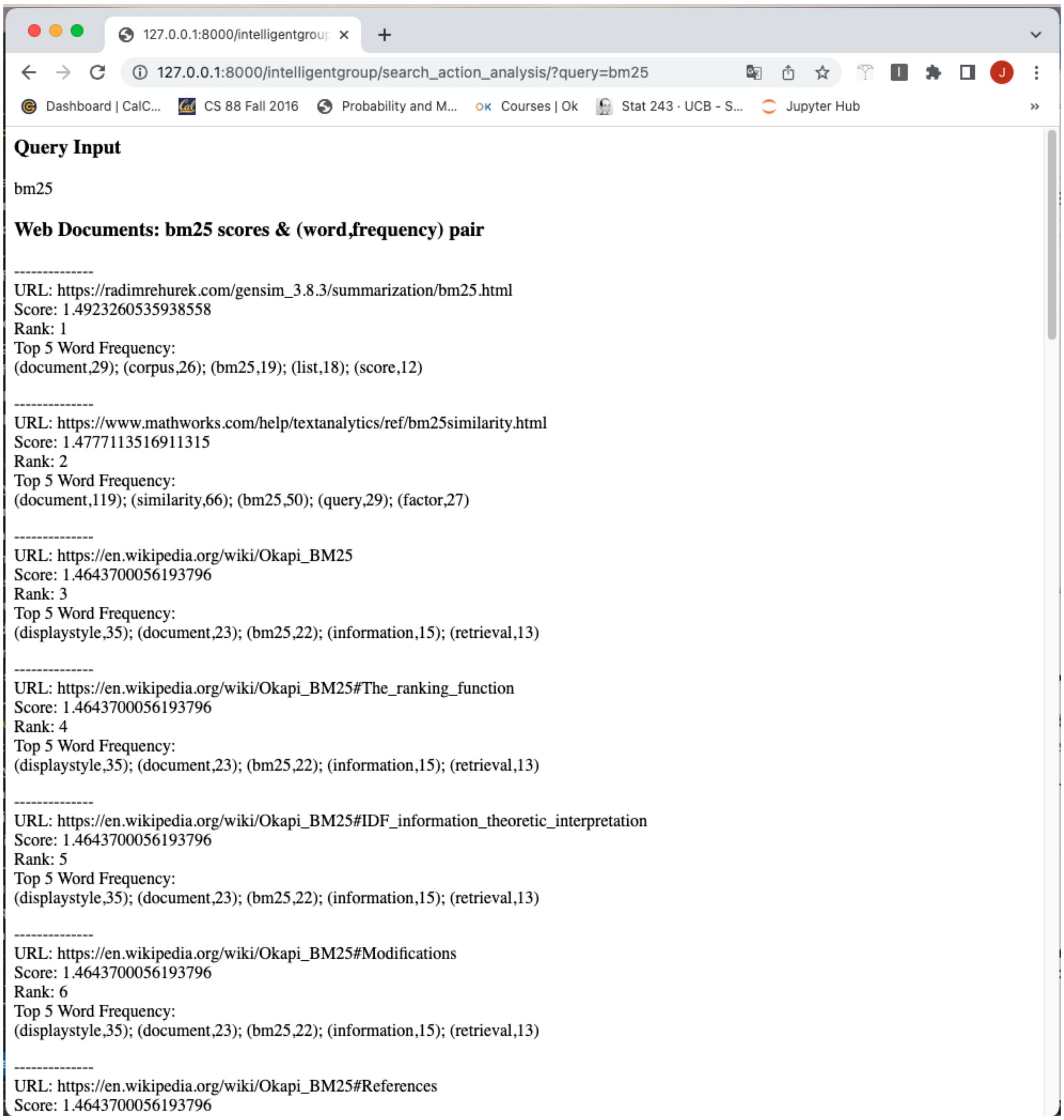
9. To use the extension, click the extension icon at the right of the address bar, then click the extension "Intelligent Search 1.0". The UI of the extension should show up.



10. Enter the query you want to analyze, then click analyze.
11. A new Chrome tab will pop up with the URL http://127.0.0.1:8000/intelligentgroup/search_action_analysis/?query=<query>, wait around 30 seconds while the backend script is running. An example output for the query term "bm25" is attached below.

Output Example

URL: http://127.0.0.1:8000/intelligentgroup/search_action_analysis/?query=bm25

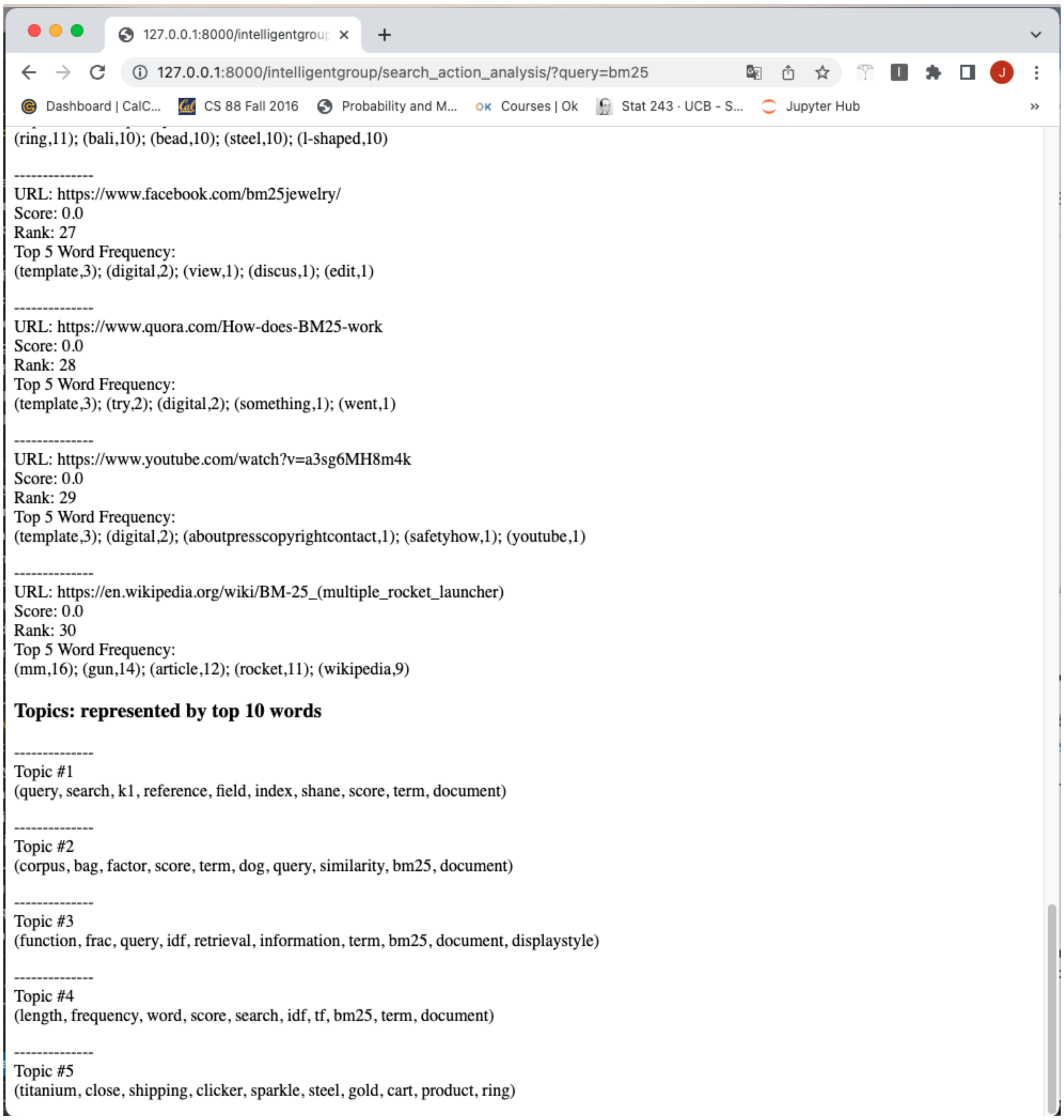


Query Input

bm25

Web Documents: bm25 scores & (word,frequency) pair

```
--------------
URL: https://radimrehurek.com/gensim_3.8.3/summarization/bm25.html
Score: 1.4923260535938558
Rank: 1
Top 5 Word Frequency:
(document,29); (corpus,26); (bm25,19); (list,18); (score,12)

--------------
URL: https://www.mathworks.com/help/textanalytics/ref/bm25similarity.html
Score: 1.4777113516911315
Rank: 2
Top 5 Word Frequency:
(document,119); (similarity,66); (bm25,50); (query,29); (factor,27)

--------------
URL: https://en.wikipedia.org/wiki/Okapi_BM25
Score: 1.4643700056193796
Rank: 3
Top 5 Word Frequency:
(displaystyle,35); (document,23); (bm25,22); (information,15); (retrieval,13)

--------------
URL: https://en.wikipedia.org/wiki/Okapi_BM25#The_ranking_function
Score: 1.4643700056193796
Rank: 4
Top 5 Word Frequency:
(displaystyle,35); (document,23); (bm25,22); (information,15); (retrieval,13)

--------------
URL: https://en.wikipedia.org/wiki/Okapi_BM25#IDF_information_theoretic_interpretation
Score: 1.4643700056193796
Rank: 5
Top 5 Word Frequency:
(displaystyle,35); (document,23); (bm25,22); (information,15); (retrieval,13)

--------------
URL: https://en.wikipedia.org/wiki/Okapi_BM25#Modifications
Score: 1.4643700056193796
Rank: 6
Top 5 Word Frequency:
(displaystyle,35); (document,23); (bm25,22); (information,15); (retrieval,13)

--------------
URL: https://en.wikipedia.org/wiki/Okapi_BM25#References
Score: 1.4643700056193796
```



```
(ring,11); (bali,10); (bead,10); (steel,10); (l-shaped,10)

--------------
URL: https://www.facebook.com/bm25jewelry/
Score: 0.0
Rank: 27
Top 5 Word Frequency:
(template,3); (digital,2); (view,1); (discus,1); (edit,1)

--------------
URL: https://www.quora.com/How-does-BM25-work
Score: 0.0
Rank: 28
Top 5 Word Frequency:
(template,3); (try,2); (digital,2); (something,1); (went,1)

--------------
URL: https://www.youtube.com/watch?v=a3sg6MH8m4k
Score: 0.0
Rank: 29
Top 5 Word Frequency:
(template,3); (digital,2); (aboutpresscopyrightcontact,1); (safetyhow,1); (youtube,1)

--------------
URL: https://en.wikipedia.org/wiki/BM-25_(multiple_rocket_launcher)
Score: 0.0
Rank: 30
Top 5 Word Frequency:
(mm,16); (gun,14); (article,12); (rocket,11); (wikipedia,9)
```

Topics: represented by top 10 words

```
--------------
Topic #1
(query, search, k1, reference, field, index, shane, score, term, document)

--------------
Topic #2
(corpus, bag, factor, score, term, dog, query, similarity, bm25, document)

--------------
Topic #3
(function, frac, query, idf, retrieval, information, term, bm25, document, displaystyle)

--------------
Topic #4
(length, frequency, word, score, search, idf, tf, bm25, term, document)

--------------
Topic #5
(titanium, close, shipping, clicker, sparkle, steel, gold, cart, product, ring)
```

From the example above, we grab the first 30 web documents from the google search with query input “bm25”, rerank them by the BM25 scores in descending order, and provide the top 5 most frequently used words in each document. Also we provide the LDA topic modeling results at the end, each topic is represented by the top 10 words in the topic.

At the first glance, the documents with higher rank in score are more relevant to the BM25 algorithm. We also noticed that a jewelry store has the same name “bm25”. The web documents related to the jewelry are assigned with a lower rank (rank 26 & 27). Web pages with errors (rank 28) or less text information (rank 29) are also assigned with lower rank.

From the topic modeling result, we successfully identify a topic(topic #5) related to the jewelry industry.

4) Brief description of contribution of each team member in case of a multi-person team.

Ali)

- Project Proposal
- Project Progress report
- Implement frontend javascript and manifest.json
- ReadMe
- Documentation
- Once files were ready, how to add figured out how to add chrome extension

Jingtian)

- Implement Python backend functionalities, including bm25 scores, word frequency calculation and LDA topic modeling
- Set up Django application server
- Implement Chrome Extension
- Project Proposal
- Project Progress report
- ReadMe
- Documentation