CS410 Text Information System

Fall 2021

Bingle - Highlight Search Engine

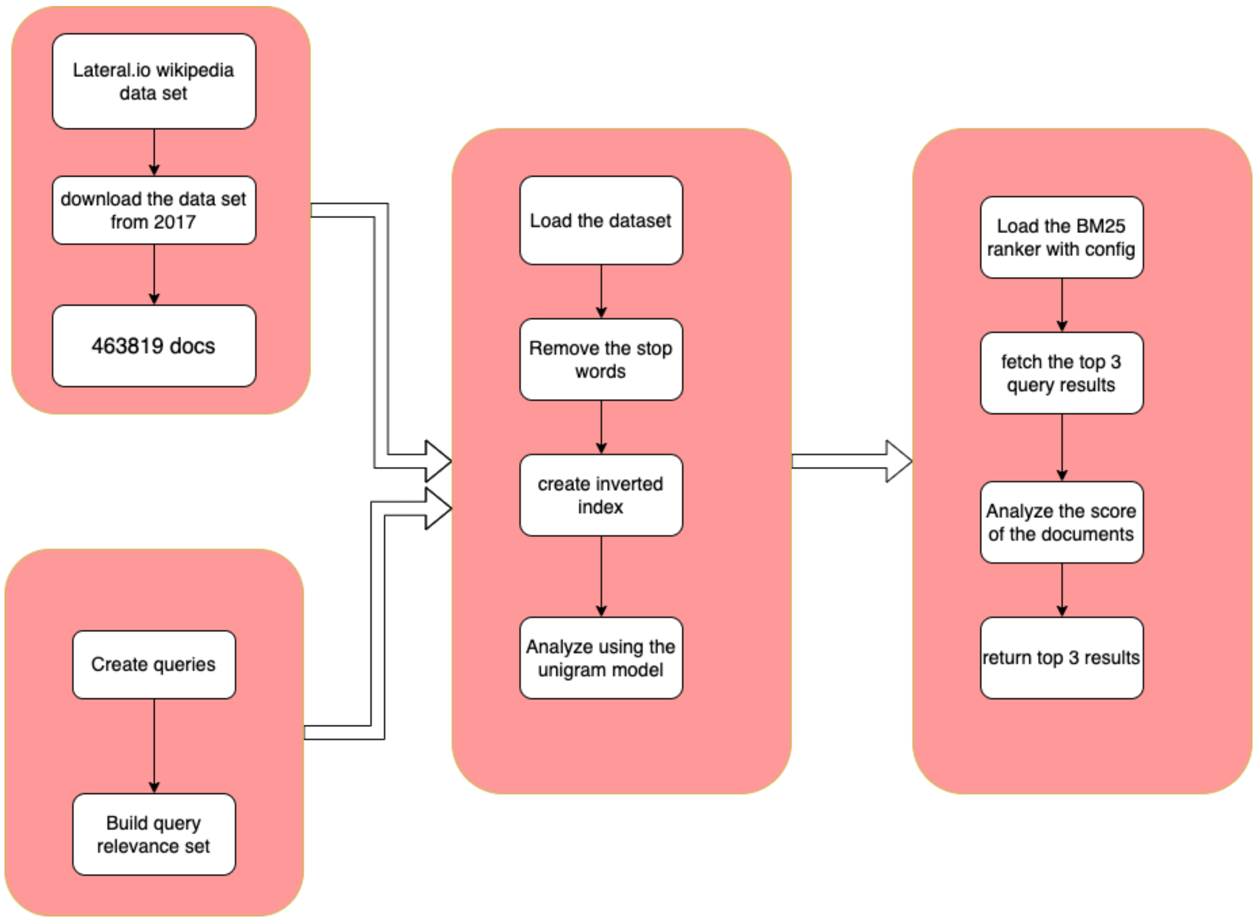
By: Shubha Sundar, Sushma Ponna, Gene Horecka, Mony Chhen

1. Introduction

Bingle is a Google Chrome extension that allows users to quickly access relevant information about some word(s) on a webpage without having to open another browser tab to search for the query. This is done by highlighting the candidate word(s) to search for, right-clicking on the highlighted word(s), and then selecting 'Search on Bingle'.

The highlighted words act as a query that will be fed into a BM25 retrieval algorithm. After the retrieval completes, Bingle will select from these results the 3 most relevant Wikipedia articles and their corresponding summaries and display them to the user in a mini-card overlay located at the top right corner of the page.

2. Data Flow diagram


Lateral.io wikipedia data set
download the data set from 2017
463819 docs
Create queries
Build query relevance set
Load the dataset
Remove the stop words
create inverted index
Analyze using the unigram model
Load the BM25 ranker with config
fetch the top 3 query results
Analyze the score of the documents
return top 3 results

3. Tasks

Tasks	
Frontend	Chrome App Extension (React.js Library + TypeScript, Deploying to Google Chrome Web Store) - ***Gene Horecka***
Backend	API Backend Server Infrastructure (Node.js Platform + Express.js Web Framework) - ***Gene Horecka***
	Information Retrieval Algorithm Development (BM25 --- searchWiki.py) - ***Shubha Sundar***
	Intelligent Search Algorithm Development (Query Expansion, NER, BM25 ---- intelligentSearch.py) ***- Sushma Ponna***
	Creating a query set & ranking evaluation - ***Shubha Sundar, Mony Chhen***
Data Selection/Storage	Curation of Data Set - ***Mony Chhen, Sushma Ponna***
Algorithm Research	Exploratory Analysis of Data Set - ***Mony Chhen, Sushma Ponna***
Documentation/Presentation	**All Team Members**

4. Dataset

Download Dataset

The Wikidata set is very large, make sure that your PC has a large enough capacity to download this. To download the dataset please follow the steps below.

Step 1: Go to https://dumps.wikimedia.org/wikidatawiki/
Step 2: Click on the latest data set you would like to download
Step 3: Save .bz2 file into local directory for parsing into .csv file

Extract Dataset

Inorder to be able to read the dataset from wikimedia parsing is required. The following steps will go over how to parse .bz2 into .csv file.
Step 1: Download git project from

➔ https://github.com/attardi/wikiextractor

Step 2: Open visual studio and open project folder
Step 3: Make sure to download the .bz2 file from the download dataset section above
Step 4: Install requirement component by running one of the two commands:

➔ Python setup.py install Or pip install wikiextractor

Step 5: Copy .bz2 file to the same folder as the wikiextrator.py file
Step 6: Run command for extracting the .bz2 file

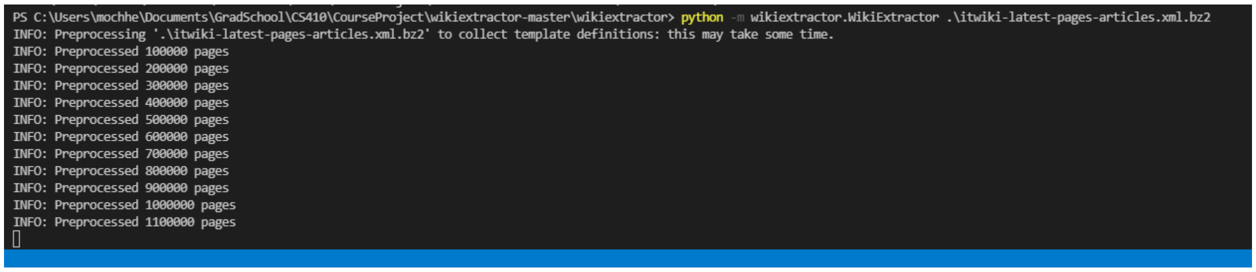
➔ python -m wikiextractor.WikiExtractor .\itwiki-latest-pages-articles.xml.bz2

Downloadable Wikipedia Dataset for the Project

The project requires the Wikipedia dataset to be accessible from within the project environment in order to run properly. These files are large and cannot be hosted on Github. Thus, we uploaded these files in a publicly accessible folder here:
https://drive.google.com/drive/folders/1zGl2SW0cdIEKgAsRAn9Jzndj08l-oasg

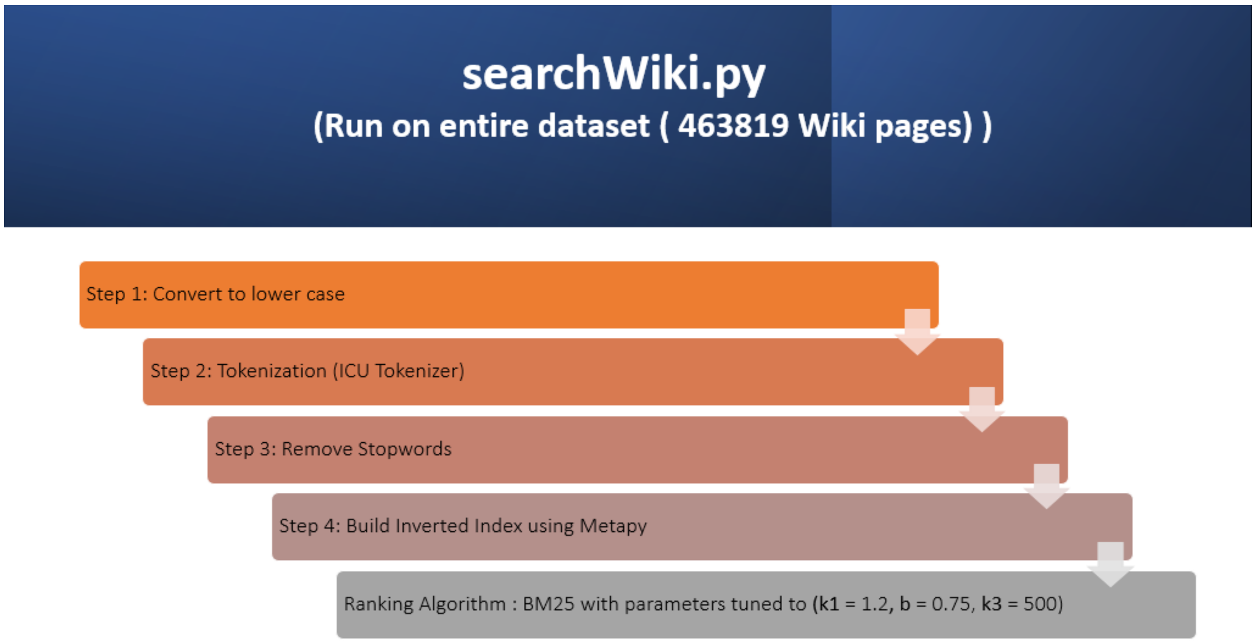
The instructions for incorporating these files with the project is outlined here:
https://github.com/genefever/bingle#download-and-add-the-necessary-index-folder-and-files-to-the-corresponding-directories

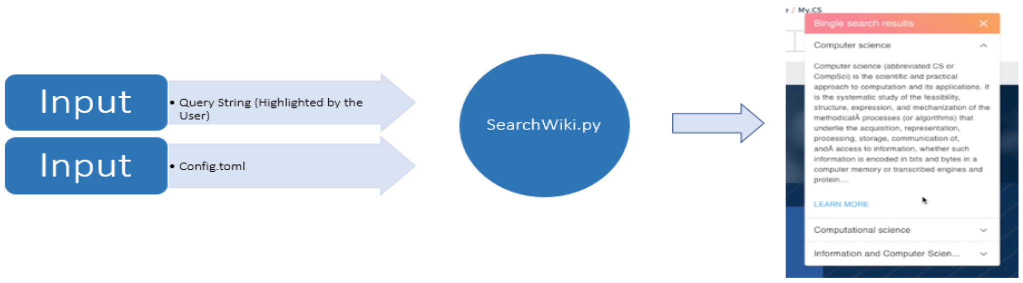
Screenshot of Step 6 running output:



```
PS C:\Users\mochhe\Documents\GradSchool\CS410\CourseProject\wikiextractor-master\wikiextractor> python -m wikiextractor.WikiExtractor .\itwiki-latest-pages-articles.xml.bz2
INFO: Preprocessing '.\itwiki-latest-pages-articles.xml.bz2' to collect template definitions: this may take some time.
INFO: Preprocessed 100000 pages
INFO: Preprocessed 200000 pages
INFO: Preprocessed 300000 pages
INFO: Preprocessed 400000 pages
INFO: Preprocessed 500000 pages
INFO: Preprocessed 600000 pages
INFO: Preprocessed 700000 pages
INFO: Preprocessed 800000 pages
INFO: Preprocessed 900000 pages
INFO: Preprocessed 1000000 pages
INFO: Preprocessed 1100000 pages
```

5. IR Algorithm Flow & Steps


searchWiki.py
(Run on entire dataset (463819 Wiki pages))
Step 1: Convert to lower case
Step 2: Tokenization (ICU Tokenizer)
Step 3: Remove Stopwords
Step 4: Build Inverted Index using Metapy
Ranking Algorithm : BM25 with parameters tuned to (k1 = 1.2, b = 0.75, k3 = 500)



Computer science

Computer science (abbreviated CS or CompSci) is the scientific and practical approach to computation and its applications. It is the systematic study of the feasibility, structure, expression, and mechanization of the methodicalÂ processes (or algorithms) that underlie the acquisition, representation, processing, storage, communication of, andÂ access to information, whether such information is encoded in bits and bytes in a computer memory or transcribed engines and protein....

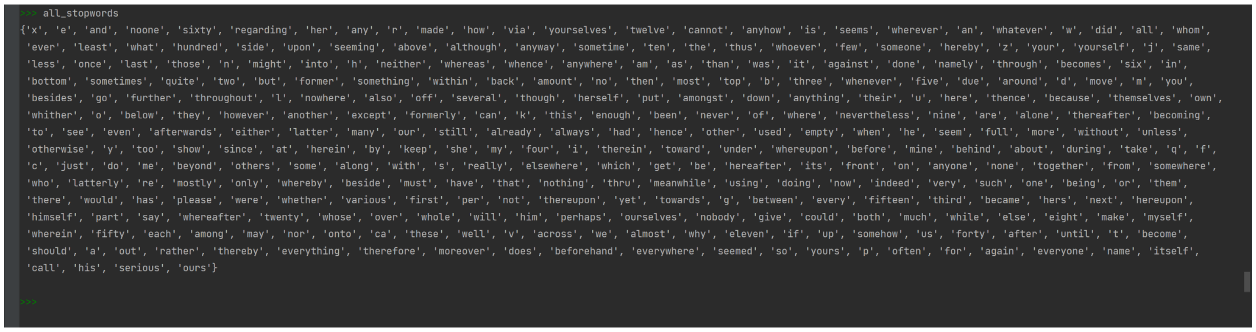
LEARN MORE

Computational science

Information and Computer Scien...

searchWiki.py is the algorithm that is currently running on Bingle with a corpus of 463819 Wikipedia pages.

Stopwords for the Stopword Removal Step



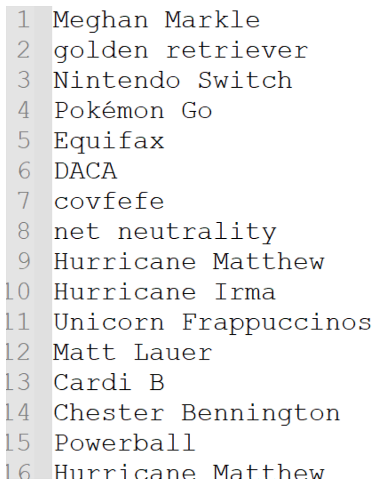
```
>>> all_stopwords
{'x', 'e', 'and', 'noone', 'sixty', 'regarding', 'her', 'any', 'r', 'made', 'how', 'via', 'yourselves', 'twelve', 'cannot', 'anyhow', 'is', 'seems', 'wherever', 'an', 'whatever', 'w', 'did', 'all', 'whom',
 'ever', 'least', 'what', 'hundred', 'side', 'upon', 'seeming', 'above', 'although', 'anyway', 'sometime', 'ten', 'the', 'thus', 'whoever', 'few', 'someone', 'hereby', 'z', 'your', 'yourself', 'j', 'same',
 'less', 'once', 'last', 'those', 'n', 'might', 'into', 'h', 'neither', 'whereas', 'whence', 'anywhere', 'am', 'as', 'than', 'was', 'it', 'against', 'done', 'namely', 'through', 'becomes', 'six', 'in',
 'bottom', 'sometimes', 'quite', 'two', 'but', 'former', 'something', 'within', 'back', 'amount', 'no', 'then', 'most', 'top', 'b', 'three', 'whenever', 'five', 'due', 'around', 'd', 'move', 'm', 'you',
 'besides', 'go', 'further', 'throughout', 'l', 'nowhere', 'also', 'off', 'several', 'though', 'herself', 'put', 'amongst', 'down', 'anything', 'their', 'u', 'here', 'thence', 'because', 'themselves', 'own',
 'whither', 'o', 'below', 'they', 'however', 'another', 'except', 'formerly', 'can', 'k', 'this', 'enough', 'been', 'never', 'of', 'where', 'nevertheless', 'nine', 'are', 'alone', 'thereafter', 'becoming',
 'to', 'see', 'even', 'afterwards', 'either', 'latter', 'many', 'our', 'still', 'already', 'always', 'had', 'hence', 'other', 'used', 'empty', 'when', 'he', 'seem', 'full', 'more', 'without', 'unless',
 'otherwise', 'y', 'too', 'show', 'since', 'at', 'herein', 'by', 'keep', 'she', 'my', 'four', 'i', 'therein', 'toward', 'under', 'whereupon', 'before', 'mine', 'behind', 'about', 'during', 'take', 'q', 'f',
 'c', 'just', 'do', 'me', 'beyond', 'others', 'some', 'along', 'with', 's', 'really', 'elsewhere', 'which', 'get', 'be', 'hereafter', 'its', 'front', 'on', 'anyone', 'none', 'together', 'from', 'somewhere',
 'who', 'latterly', 're', 'mostly', 'only', 'whereby', 'beside', 'must', 'have', 'that', 'nothing', 'thru', 'meanwhile', 'using', 'doing', 'now', 'indeed', 'very', 'such', 'one', 'being', 'or', 'them',
 'there', 'would', 'has', 'please', 'were', 'whether', 'various', 'first', 'per', 'not', 'thereupon', 'yet', 'towards', 'g', 'between', 'every', 'fifteen', 'third', 'became', 'hers', 'next', 'hereupon',
 'himself', 'part', 'say', 'whereafter', 'twenty', 'whose', 'over', 'whole', 'will', 'him', 'perhaps', 'ourselves', 'nobody', 'give', 'could', 'both', 'much', 'while', 'else', 'eight', 'make', 'myself',
 'wherein', 'fifty', 'each', 'among', 'may', 'nor', 'onto', 'ca', 'these', 'well', 'v', 'across', 'we', 'almost', 'why', 'eleven', 'if', 'up', 'somehow', 'us', 'forty', 'after', 'until', 't', 'become',
 'should', 'a', 'out', 'rather', 'thereby', 'everything', 'therefore', 'moreover', 'does', 'beforehand', 'everywhere', 'seemed', 'so', 'yours', 'p', 'often', 'for', 'again', 'everyone', 'name', 'itself',
 'call', 'his', 'serious', 'ours'}

>>>
```

6. IR Evaluation

To ensure that our search engine is working as expected, we created a test query set and manually ranked the queries and built a query relevance set for the respective queries in the query set. The relevance is ranked on a scale of 0 to 4, 4 being the most relevant document. The length of the query string is chosen to be around 1-5 words keeping in mind the application of the use case. (We assume that the average user would highlight a word or a short phrase). The IR algorithm is evaluated against this 50 manually created query set. The IR algorithm is set to output the top 3 ranked search results which are evaluated by calculating Average Precision of each query with the ground truth from the

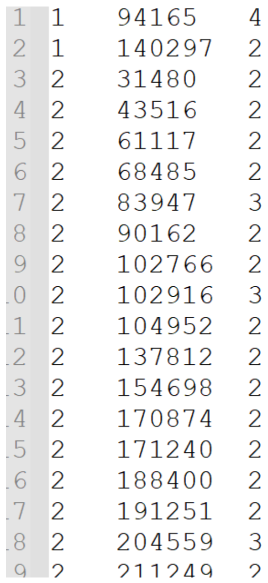
Query relevance test set. Mean Average Precision which is the average of AP of all the queries is calculated, which is an indicator of how good the search algorithm performed on the test set.

A glimpse of the Test Query Set.



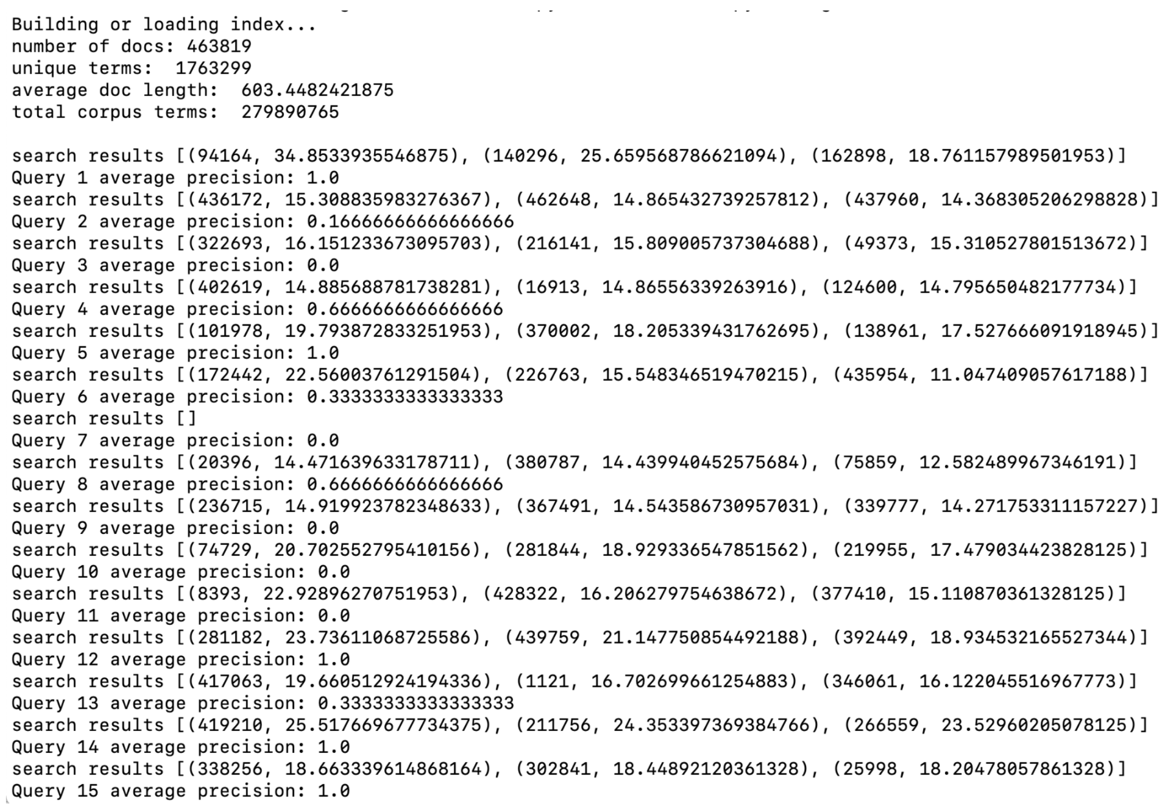
```
Meghan Markle
golden retriever
Nintendo Switch
Pokémon Go
Equifax
DACA
covfefe
net neutrality
Hurricane Matthew
Hurricane Irma
Unicorn Frappuccinos
Matt Lauer
Cardi B
Chester Bennington
Powerball
Hurricane Matthew
```

A glimpse of the Test Query Relevance set. (Format: Query-id, Doc-id, Relevance)



```
1    94165    4
1    140297   2
2    31480    2
2    43516    2
2    61117    2
2    68485    2
2    83947    3
2    90162    2
2    102766   2
2    102916   3
2    104952   2
2    137812   2
2    154698   2
2    170874   2
2    171240   2
2    188400   2
2    191251   2
2    204559   3
2    211249   2
```

Performance of the IR algorithm on the test set:



```
Building or loading index...
number of docs: 463819
unique terms:  1763299
average doc length:  603.4482421875
total corpus terms:  279890765

search results [(94164, 34.8533935546875), (140296, 25.659568786621094), (162898, 18.761157989501953)]
Query 1 average precision: 1.0
search results [(436172, 15.308835983276367), (462648, 14.865432739257812), (437960, 14.368305206298828)]
Query 2 average precision: 0.16666666666666666
search results [(322693, 16.151233673095703), (216141, 15.809005737304688), (49373, 15.310527801513672)]
Query 3 average precision: 0.0
search results [(402619, 14.885688781738281), (16913, 14.86556339263916), (124600, 14.795650482177734)]
Query 4 average precision: 0.6666666666666666
search results [(101978, 19.793872833251953), (370002, 18.205339431762695), (138961, 17.527666091918945)]
Query 5 average precision: 1.0
search results [(172442, 22.56003761291504), (226763, 15.548346519470215), (435954, 11.047409057617188)]
Query 6 average precision: 0.3333333333333333
search results []
Query 7 average precision: 0.0
search results [(20396, 14.471639633178711), (380787, 14.439940452575684), (75859, 12.582489967346191)]
Query 8 average precision: 0.6666666666666666
search results [(236715, 14.919923782348633), (367491, 14.543586730957031), (339777, 14.271753311157227)]
Query 9 average precision: 0.0
search results [(74729, 20.702552795410156), (281844, 18.929336547851562), (219955, 17.479034423828125)]
Query 10 average precision: 0.0
search results [(8393, 22.92896270751953), (428322, 16.206279754638672), (377410, 15.110870361328125)]
Query 11 average precision: 0.0
search results [(281182, 23.73611068725586), (439759, 21.147750854492188), (392449, 18.934532165527344)]
Query 12 average precision: 1.0
search results [(417063, 19.660512924194336), (1121, 16.702699661254883), (346061, 16.122045516967773)]
Query 13 average precision: 0.3333333333333333
search results [(419210, 25.517669677734375), (211756, 24.353397369384766), (266559, 23.52960205078125)]
Query 14 average precision: 1.0
search results [(338256, 18.663339614868164), (302841, 18.44892120361328), (25998, 18.20478057861328)]
Query 15 average precision: 1.0
```

7. Chrome Extension Code Documentation

This section explains the important files that constitute the Bingle Chrome extension. It does not describe how to build and install the Chrome extension manually. For that, please refer to https://github.com/genefever/bingle/blob/main/README.md.

All the Chrome extension-related files and folders are placed under the **bingle/chrome_extension/** project directory.

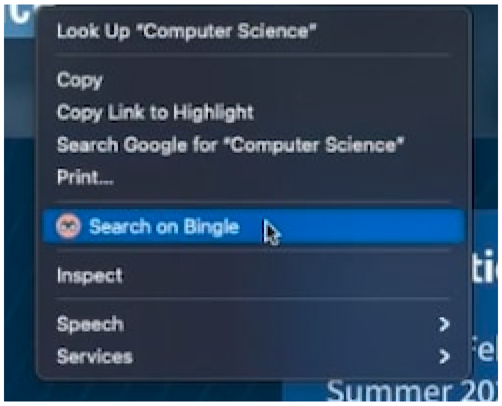
api.ts

This file contains the fetchWikiData() function that makes the HTTP GET request to the backend API server to fetch the three most relevant Wikipedia data to display to the UI. It also contains the interface 'WikiData' that defines the Wikipedia data structure that is used within the UI.

background.ts

The background script reacts to browser events and performs certain actions, such as initializing the context menu when the Chrome extension is installed or enabling message passing within the application. In our case, we use this script to inject the "Search on Bingle" button within the dropdown menu on installation.

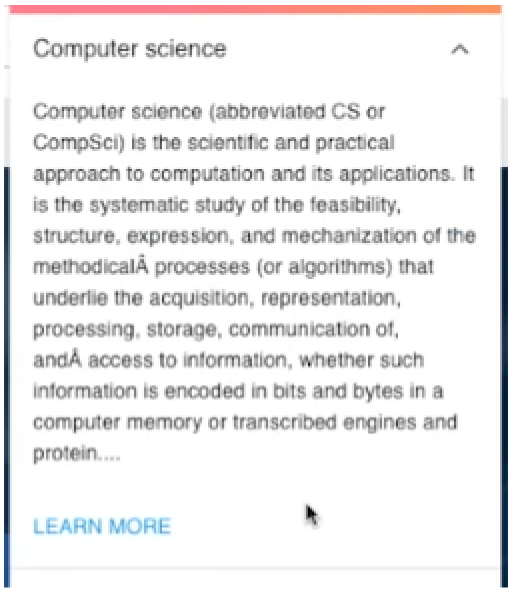
We also add listeners to react to certain events that occur during the runtime of the extension, such as sending the highlighted search query text to contentScript.tsx when the user clicks on “Search on Bingle”.



background.ts

components/InfoCard.tsx

The InfoCard is a React component that displays the title, Wikipedia article description, and “Learn More” button (linked with the corresponding Wikipedia article URL). It utilizes the [Material UI]() React library to facilitate the dropdown menu action.



Computer science

Computer science (abbreviated CS or CompSci) is the scientific and practical approach to computation and its applications. It is the systematic study of the feasibility, structure, expression, and mechanization of the methodicalÂ processes (or algorithms) that underlie the acquisition, representation, processing, storage, communication of, andÂ access to information, whether such information is encoded in bits and bytes in a computer memory or transcribed engines and protein....

LEARN MORE

InfoCard.tsx

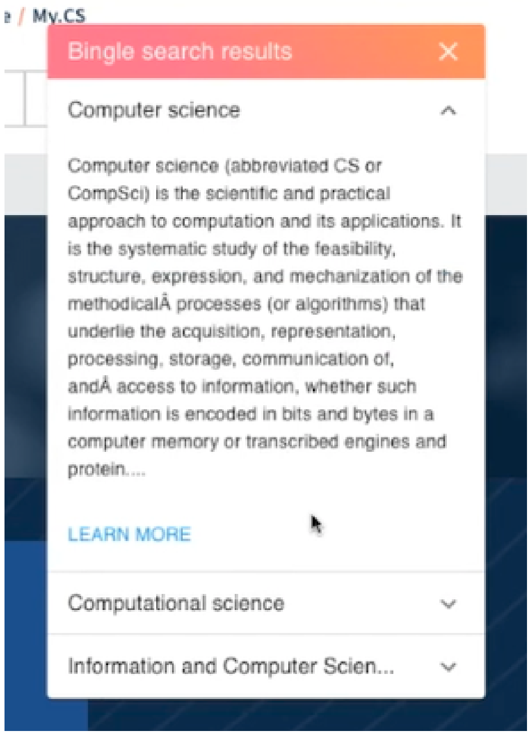
components/Popup.tsx

The Popup is a React component that contains 3 InfoCard components. Within the Popup is where the retrieved search query candidates are populated individually into each InfoCard.

contentScript.tsx

The content script is a Javascript-based file that runs in the context of web pages. This means that a content script can interact with web pages that the browser visits. Not every Javascript file in a Chrome extension can do this, so this file can be thought of as a special Chrome extension file.

This file contains the main UI code and logic behind Bingle's dropdown mini-card overlay that appears on the web page when a user clicks "Search on Bingle".



Computer science (abbreviated CS or CompSci) is the scientific and practical approach to computation and its applications. It is the systematic study of the feasibility, structure, expression, and mechanization of the methodicalÂ processes (or algorithms) that underlie the acquisition, representation, processing, storage, communication of, andÂ access to information, whether such information is encoded in bits and bytes in a computer memory or transcribed engines and protein....

contentScript.tsx

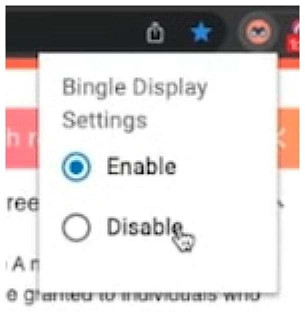
The UI component is built using React and Typescript and contains the Popup and InfoCard components described above. It utilizes the Material UI React library to facilitate the design and actions of the extension.

manifest.json

The manifest.json file tells Chrome important information about the extension, such as its name, version, and the permissions that it needs.

popup.tsx

This React component is different from the Popup.tsx file mentioned above (as differentiated by the first-letter 'p' case difference). This React component shows the "Bingle Display Settings'' options popup. It is named popup.tsx because it is a special file that is picked up by manifest.json, which instructs it to display this UI right under the extension icon in the Chrome extension taskbar. This UI component is only displayed when a user clicks on the Bingle extension icon.



popup.tsx

This file also contains logic that talks to the browser's local storage (storage.ts) to store the user's Bingle Display Settings.

storage.ts

This file contains the code that stores user information from the extension on the Chrome browser's local storage. The extension currently only stores information about the extension display settings (enable/disable extension).

tsconfig.json

The tsconfig.json file allows you to specify the root level files and the compiler options that are required to compile a TypeScript project. This is needed because this extension is built using TypeScript. The presence of this file in the project directory specifies that the said directory is the TypeScript project root.

types.ts

The types.ts file contains custom type definitions for different objects used in this project. It currently contains type definitions for message passing logic and event handling logic.

webpack.common.js

We use webpack to bundle our Chrome extension project. The main purpose of webpack is to bundle Javascript files for usage in a browser, but we also use it to transform other front-end assets, such as our CSS files and icon image. The webpack.common.js file contains the information needed by webpack to bundle our project so that it can run on the Chrome browser.

8. Node.js API Server Code Documentation

We use Node.js as our backend server environment and use the python-shell npm library to run our Python scripts from Node.js. We decided to go with Node.js as our backend platform because we had previous developer knowledge about this environment.

The API server receives an HTTP GET request from the Chrome extension that contains the search query terms to retrieve information about. The server processes this request and responds with the retrieved information back to the client to display on the Chrome extension.

app.js
The app.js file is the initial file of this Node.js application. It contains the code and middleware libraries to create and run the server. We use Express.js as our web application framework for Node.js. We also include routing paths from within app.js.

api.routes.js
This file contains the routing information needed by Express.js to route the client requests to the processing logic. We keep this routing file separate from app.js so that the code is more modular and enables better scalability for future development.

api.controllers.js
This file contains the request processing logic. It is here where Node.js invokes the searchWiki.py Python script to run our retrieval algorithm. The getRankedWikiData() function is what parses the incoming request's query parameter and passes them to the Python script.

Once the Python script finishes and returns the retrieved information as a JSON object, the getRankedWikiData() function packages this JSON object and sends it back as a response to the client.

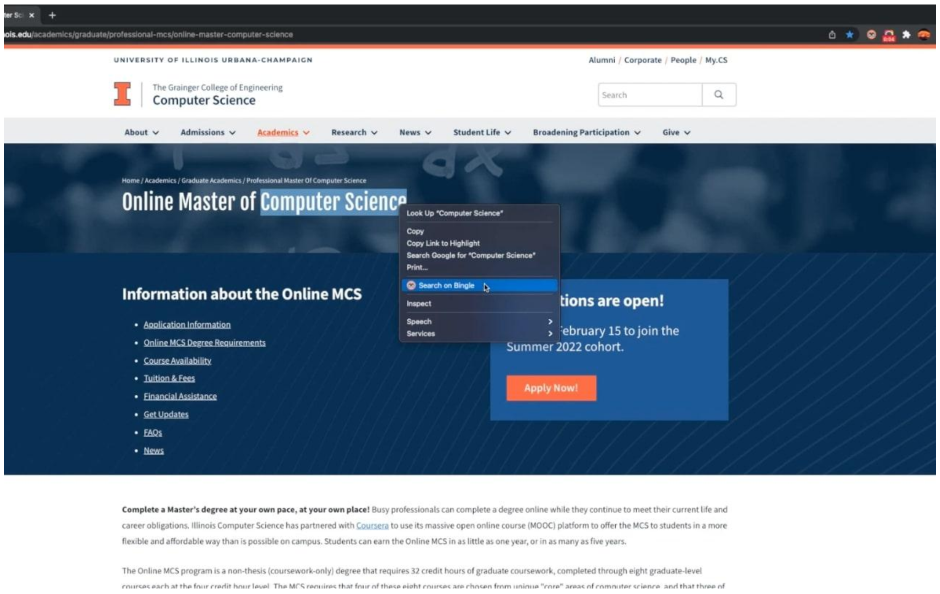
9. Bingle User Guide

Project installation and user guide: https://github.com/genefever/bingle/blob/main/README.md

Video tutorial: https://www.youtube.com/watch?v=7Xv4ZQc5zuk

Usage:

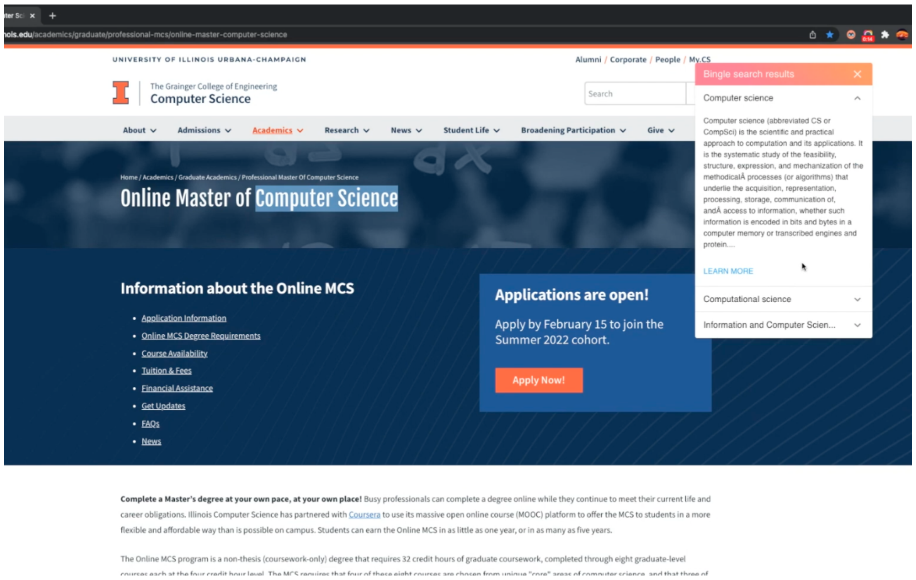
Step 1: To activate, highlight some word(s) on a webpage and right-click on the highlighted word(s), then select "Search on Bingle" from the dropdown menu. The highlighted word(s) act as the search query.


UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Alumni / Corporate / People / My.CS
The Grainger College of Engineering
Computer Science
Search
About
Admissions
Academics
Research
News
Student Life
Broadening Participation
Give
Home / Academics / Graduate Academics / Professional Master Of Computer Science
Online Master of Computer Science
Look Up "Computer Science"
Copy
Copy Link to Highlight
Search Google for "Computer Science"
Print...
Search on Bingle
Inspect
Speech
Services
Information about the Online MCS
Application Information
Online MCS Degree Requirements
Course Availability
Tuition & Fees
Financial Assistance
Get Updates
FAQs
News
tions are open!
ebruary 15 to join the Summer 2022 cohort.
Apply Now!

Complete a Master's degree at your own pace, at your own place! Busy professionals can complete a degree online while they continue to meet their current life and career obligations. Illinois Computer Science has partnered with Coursera to use its massive open online course (MOOC) platform to offer the MCS to students in a more flexible and affordable way than is possible on campus. Students can earn the Online MCS in as little as one year, or in as many as five years.

The Online MCS program is a non-thesis (coursework-only) degree that requires 32 credit hours of graduate coursework, completed through eight graduate-level

Step 2: Bingle will return the 3 most relevant search results based on the search query. The results are displayed in a mini dropdown list at the top right corner of your Chrome web browser, as shown in the screenshot below. Each dropdown card will contain a summary of the most relevant search results taken from Wikipedia, along with a corresponding link to the Wikipedia article.

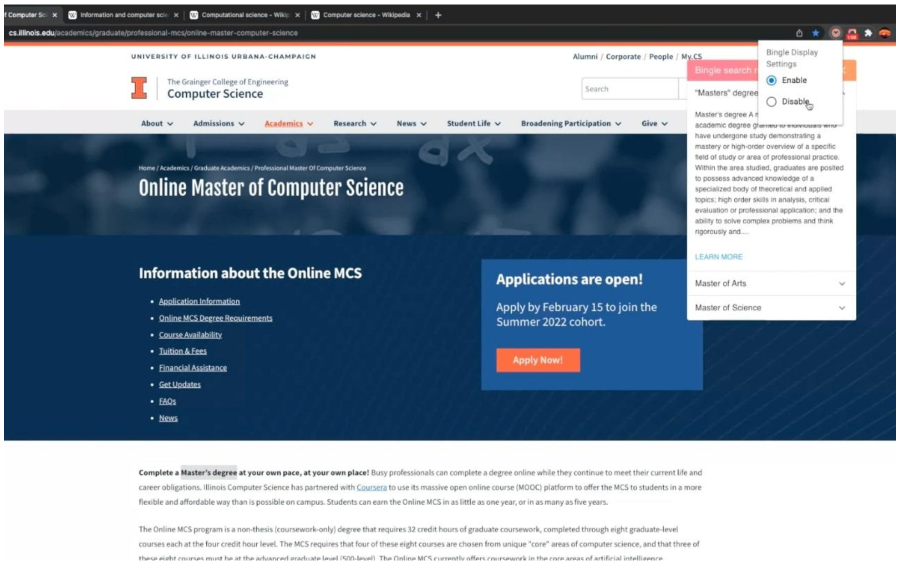


Computer science (abbreviated CS or CompSci) is the scientific and practical approach to computation and its applications. It is the systematic study of the feasibility, structure, expression, and mechanization of the methodicalÂ processes (or algorithms) that underlie the acquisition, representation, processing, storage, communication of, andÂ access to information, whether such information is encoded in bits and bytes in a computer memory or transcribed engines and protein....

Complete a Master's degree at your own pace, at your own place! Busy professionals can complete a degree online while they continue to meet their current life and career obligations. Illinois Computer Science has partnered with Coursera to use its massive open online course (MOOC) platform to offer the MCS to students in a more flexible and affordable way than is possible on campus. Students can earn the Online MCS in as little as one year, or in as many as five years.

The Online MCS program is a non-thesis (coursework-only) degree that requires 32 credit hours of graduate coursework, completed through eight graduate-level

Step 3: You can enable/disable the Bingle Chrome extension by clicking on the Bingle icon on the Chrome extension taskbar located at the top right corner of your browser. Disabling the extension will remove "Search from Bingle" from the dropdown menu.


Computer Sc
Information and computer scie
Computational science - Wikip
Computer science - Wikipedia
cs.illinois.edu/academics/graduate/professional-mcs/online-master-computer-science
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Alumni / Corporate / People / My.CS
The Grainger College of Engineering
Computer Science
Search
About
Admissions
Academics
Research
News
Student Life
Broadening Participation
Give
Bingle search r
LEARN MORE
Master of Arts
Master of Science
Apply Now!

Bingle Display Settings

[✓ Enable] Disable

"Masters" degree

Master's degree A r… academic degree granted to individuals who have undergone study demonstrating a mastery or high-order overview of a specific field of study or area of professional practice. Within the area studied, graduates are posited to possess advanced knowledge of a specialized body of theoretical and applied topics; high order skills in analysis, critical evaluation or professional application; and the ability to solve complex problems and think rigorously and....

Home / Academics / Graduate Academics / Professional Master Of Computer Science

Online Master of Computer Science

Information about the Online MCS

- Application Information
- Online MCS Degree Requirements
- Course Availability
- Tuition & Fees
- Financial Assistance
- Get Updates
- FAQs
- News

Applications are open!

Apply by February 15 to join the Summer 2022 cohort.

Complete a Master's degree at your own pace, at your own place! Busy professionals can complete a degree online while they continue to meet their current life and career obligations. Illinois Computer Science has partnered with Coursera to use its massive open online course (MOOC) platform to offer the MCS to students in a more flexible and affordable way than is possible on campus. Students can earn the Online MCS in as little as one year, or in as many as five years.

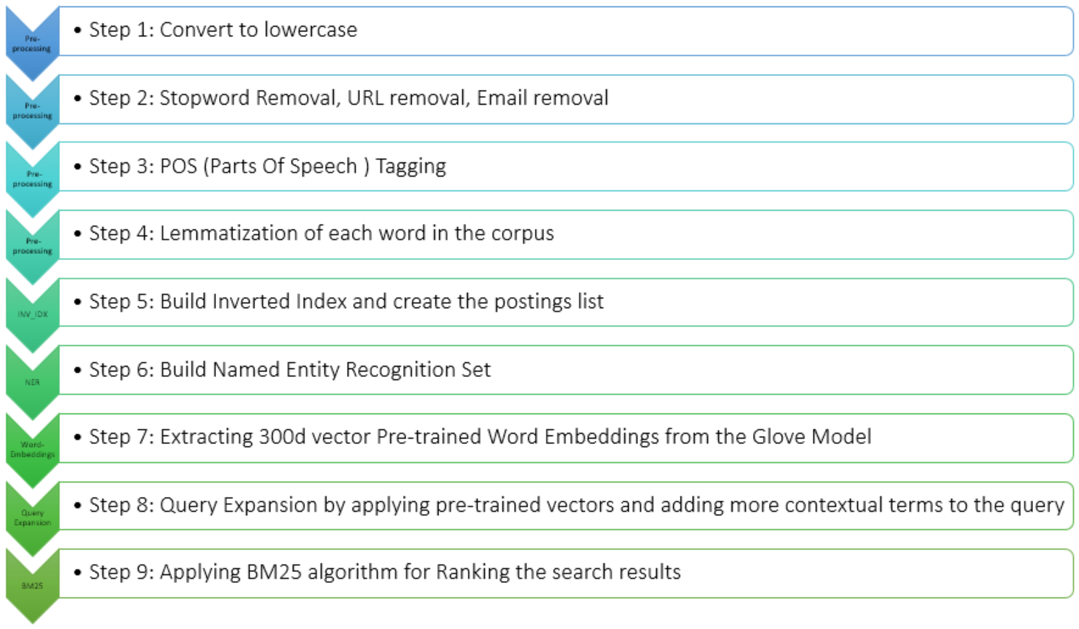
The Online MCS program is a non-thesis (coursework-only) degree that requires 32 credit hours of graduate coursework, completed through eight graduate-level courses each at the four credit hour level. The MCS requires that four of these eight courses are chosen from unique "core" areas of computer science, and that three of these eight courses must be at the advanced graduate level (500-level). The Online MCS currently offers coursework in the core areas of artificial intelligence

10. Algorithm Research

Query Expansion + BM25

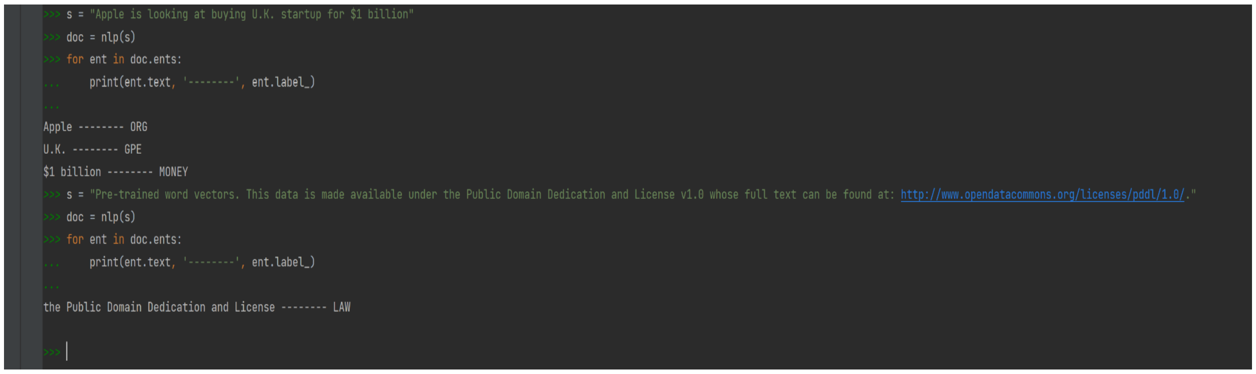
We were able to implement and test other NLP tasks such as NER, Query Expansion on a much smaller dataset ~2k Wikipedia pages(selected from the 463819 wiki pages dataset). The code for this is in ***intelligentSearch.py*** , the IR Algorithm and Flow for which is demonstrated below

intelligentSearch.py
(Run on a small dataset (1989 Wiki Pages))



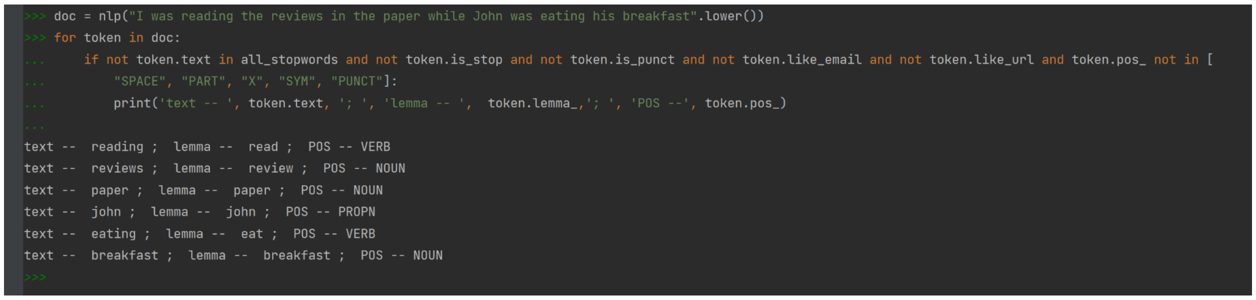
- Step 1: Convert to lowercase
- Step 2: Stopword Removal, URL removal, Email removal
- Step 3: POS (Parts Of Speech) Tagging
- Step 4: Lemmatization of each word in the corpus
- Step 5: Build Inverted Index and create the postings list
- Step 6: Build Named Entity Recognition Set
- Step 7: Extracting 300d vector Pre-trained Word Embeddings from the Glove Model
- Step 8: Query Expansion by applying pre-trained vectors and adding more contextual terms to the query
- Step 9: Applying BM25 algorithm for Ranking the search results

Example of Named Entity Recognition (NER) using Spacy:



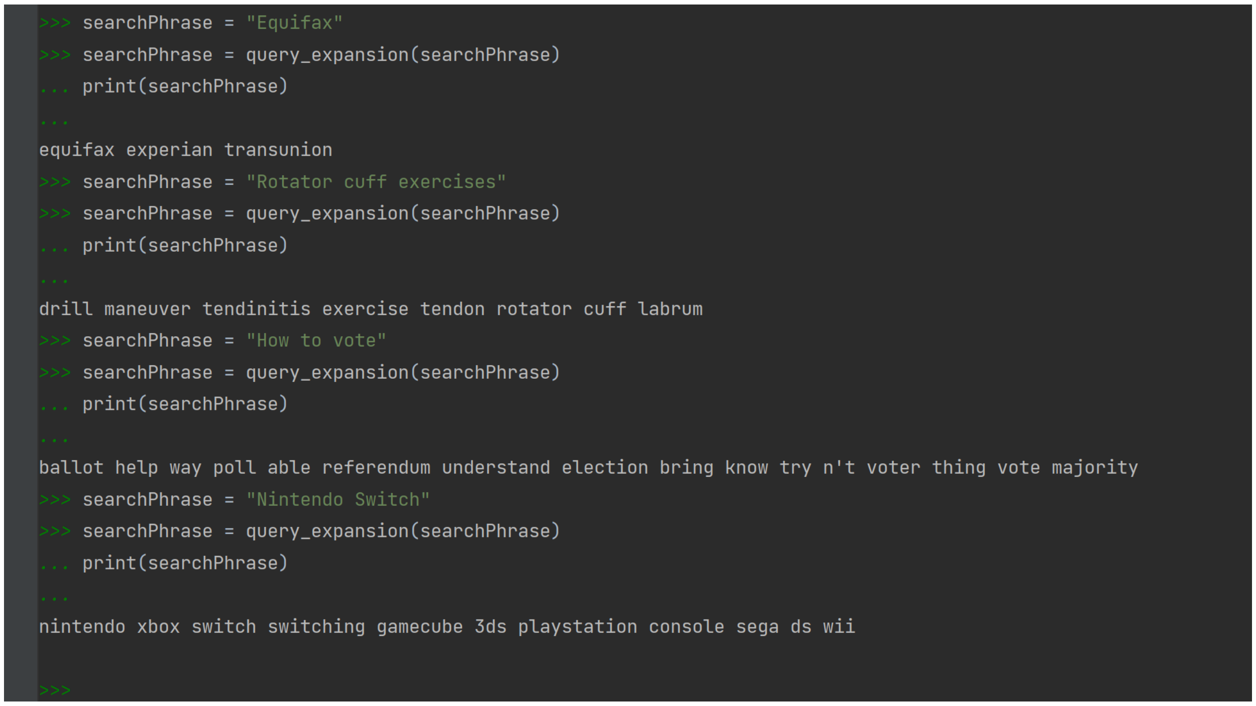
```
>>> s = "Apple is looking at buying U.K. startup for $1 billion"
>>> doc = nlp(s)
>>> for ent in doc.ents:
...     print(ent.text, '--------', ent.label_)
...
Apple -------- ORG
U.K. -------- GPE
$1 billion -------- MONEY
>>> s = "Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/."
>>> doc = nlp(s)
>>> for ent in doc.ents:
...     print(ent.text, '--------', ent.label_)
...
the Public Domain Dedication and License -------- LAW

>>> |
```

Example of Lemmatization and POS tagging



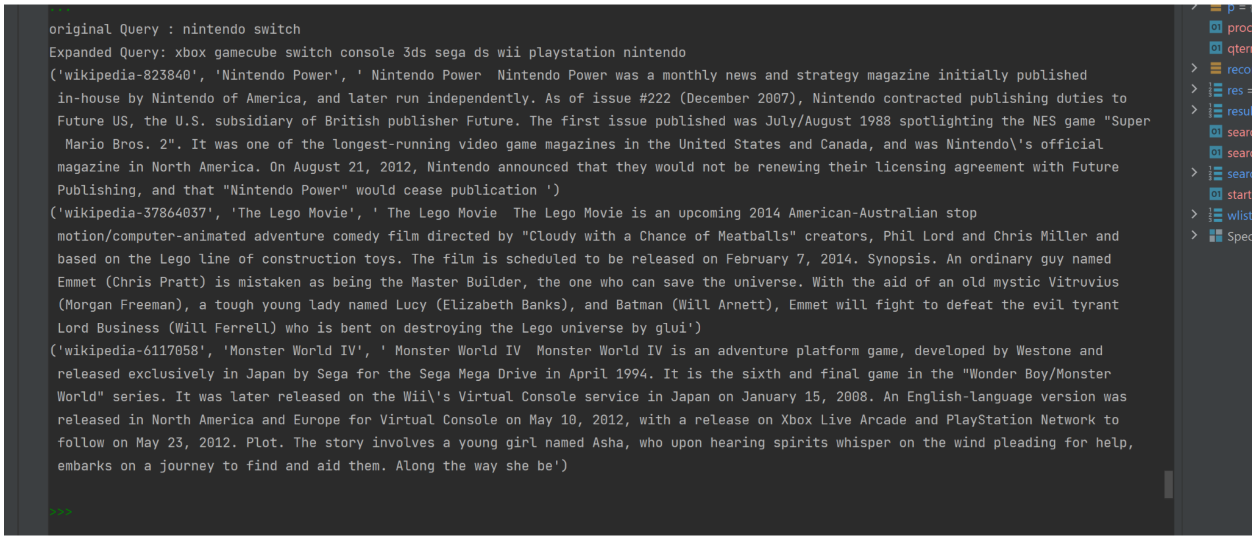
```
>>> doc = nlp("I was reading the reviews in the paper while John was eating his breakfast".lower())
>>> for token in doc:
...     if not token.text in all_stopwords and not token.is_stop and not token.is_punct and not token.like_email and not token.like_url and token.pos_ not in [
...         "SPACE", "PART", "X", "SYM", "PUNCT"]:
...         print('text -- ', token.text, '; ', 'lemma -- ',  token.lemma_,'; ', 'POS --', token.pos_)
...
text --  reading ;  lemma --  read ;  POS -- VERB
text --  reviews ;  lemma --  review ;  POS -- NOUN
text --  paper ;  lemma --  paper ;  POS -- NOUN
text --  john ;  lemma --  john ;  POS -- PROPN
text --  eating ;  lemma --  eat ;  POS -- VERB
text --  breakfast ;  lemma --  breakfast ;  POS -- NOUN
>>>
```

Example of Query Expansion:



```
>>> searchPhrase = "Equifax"
>>> searchPhrase = query_expansion(searchPhrase)
... print(searchPhrase)
...
equifax experian transunion
>>> searchPhrase = "Rotator cuff exercises"
>>> searchPhrase = query_expansion(searchPhrase)
... print(searchPhrase)
...
drill maneuver tendinitis exercise tendon rotator cuff labrum
>>> searchPhrase = "How to vote"
>>> searchPhrase = query_expansion(searchPhrase)
... print(searchPhrase)
...
ballot help way poll able referendum understand election bring know try n't voter thing vote majority
>>> searchPhrase = "Nintendo Switch"
>>> searchPhrase = query_expansion(searchPhrase)
... print(searchPhrase)
...
nintendo xbox switch switching gamecube 3ds playstation console sega ds wii

>>>
```

Sample Output of intelligentSearch.py

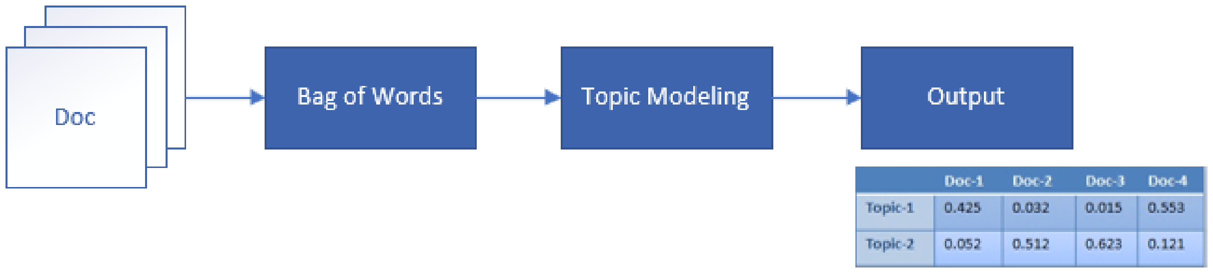


```
...
original Query : nintendo switch
Expanded Query: xbox gamecube switch console 3ds sega ds wii playstation nintendo
('wikipedia-823840', 'Nintendo Power', ' Nintendo Power  Nintendo Power was a monthly news and strategy magazine initially published
 in-house by Nintendo of America, and later run independently. As of issue #222 (December 2007), Nintendo contracted publishing duties to
 Future US, the U.S. subsidiary of British publisher Future. The first issue published was July/August 1988 spotlighting the NES game "Super
  Mario Bros. 2". It was one of the longest-running video game magazines in the United States and Canada, and was Nintendo\'s official
 magazine in North America. On August 21, 2012, Nintendo announced that they would not be renewing their licensing agreement with Future
 Publishing, and that "Nintendo Power" would cease publication ')
('wikipedia-37864037', 'The Lego Movie', ' The Lego Movie  The Lego Movie is an upcoming 2014 American-Australian stop
 motion/computer-animated adventure comedy film directed by "Cloudy with a Chance of Meatballs" creators, Phil Lord and Chris Miller and
 based on the Lego line of construction toys. The film is scheduled to be released on February 7, 2014. Synopsis. An ordinary guy named
 Emmet (Chris Pratt) is mistaken as being the Master Builder, the one who can save the universe. With the aid of an old mystic Vitruvius
 (Morgan Freeman), a tough young lady named Lucy (Elizabeth Banks), and Batman (Will Arnett), Emmet will fight to defeat the evil tyrant
 Lord Business (Will Ferrell) who is bent on destroying the Lego universe by glui')
('wikipedia-6117058', 'Monster World IV', ' Monster World IV  Monster World IV is an adventure platform game, developed by Westone and
 released exclusively in Japan by Sega for the Sega Mega Drive in April 1994. It is the sixth and final game in the "Wonder Boy/Monster
 World" series. It was later released on the Wii\'s Virtual Console service in Japan on January 15, 2008. An English-language version was
 released in North America and Europe for Virtual Console on May 10, 2012, with a release on Xbox Live Arcade and PlayStation Network to
 follow on May 23, 2012. Plot. The story involves a young girl named Asha, who upon hearing spirits whisper on the wind pleading for help,
 embarks on a journey to find and aid them. Along the way she be')

>>>
```

LSA Algorithm

LSA is an algorithm that discovers different topics from each document in the whole data set. It is a text analytics algorithm that is used for finding the group of words for each document. These groups of words represent a topic that the document talks about. There is a possibility that a document can contain more than a single topic.

We wanted to discover and add context for the documents in the corpus and experimented with LSA with the number of topics. We plan on adding this to our larger dataset in the future.
Algorithm Flow:



	Doc-1	Doc-2	Doc-3	Doc-4
Topic-1	0.425	0.032	0.015	0.553
Topic-2	0.052	0.512	0.623	0.121

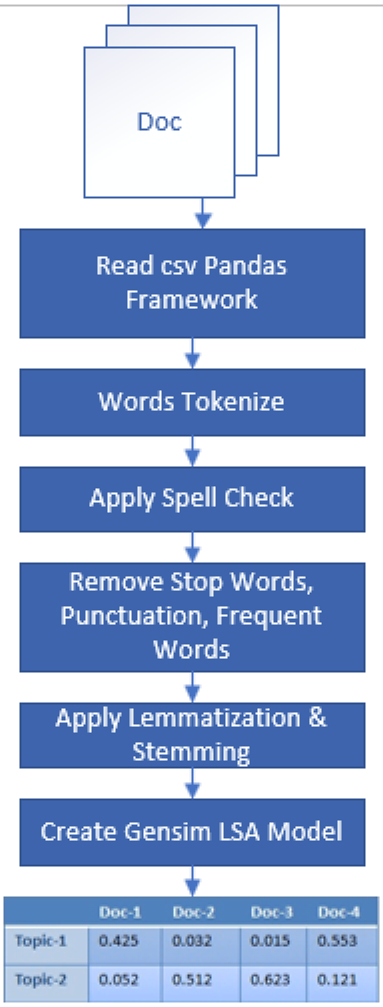
To run LSA.py, there are multiple packages that can be downloaded and installed. Follow the steps below for installing all packages required for LSA to work.

<u>Package Installation:</u>
Step 1: Run ‘pip install –upgrade gensim
- ➔ https://pypi.org/project/gensim/

Step 2: Run ‘pip install --user -U nltk’
- ➔ https://www.nltk.org/install.html

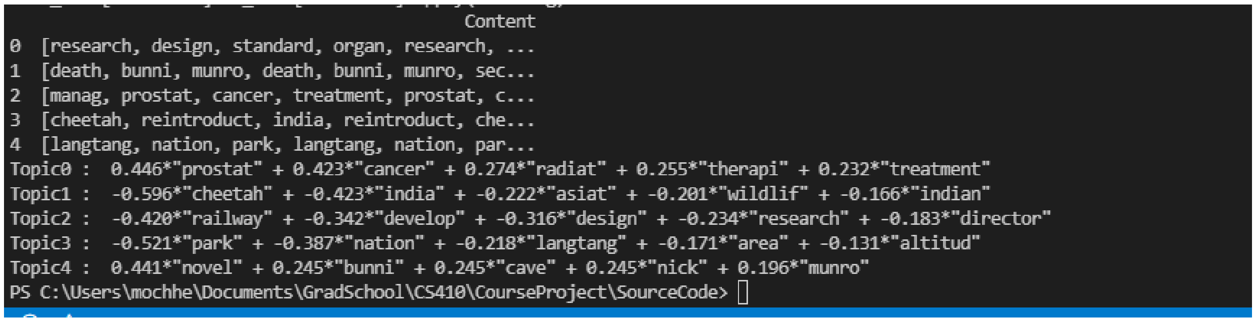
Step 3: Run ‘pip install --user -U numpy’
Step 4: Run ‘python -m nltk.downloader all’
Step 5: Run ‘pip install pyspellchecker’
Step 6: Run ‘pip install pandas’

<u>Code Flow - LSA_Chunk.py:</u>


Doc
Read csv Pandas Framework
Words Tokenize
Apply Spell Check
Remove Stop Words, Punctuation, Frequent Words
Apply Lemmatization & Stemming
Create Gensim LSA Model

	Doc-1	Doc-2	Doc-3	Doc-4
Topic-1	0.425	0.032	0.015	0.553
Topic-2	0.052	0.512	0.623	0.121

Output example with 5 Topics, and 5 words



```
                                            Content
0  [research, design, standard, organ, research, ...
1  [death, bunni, munro, death, bunni, munro, sec...
2  [manag, prostat, cancer, treatment, prostat, c...
3  [cheetah, reintroduct, india, reintroduct, che...
4  [langtang, nation, park, langtang, nation, par...
Topic0 :  0.446*"prostat" + 0.423*"cancer" + 0.274*"radiat" + 0.255*"therapi" + 0.232*"treatment"
Topic1 :  -0.596*"cheetah" + -0.423*"india" + -0.222*"asiat" + -0.201*"wildlif" + -0.166*"indian"
Topic2 :  -0.420*"railway" + -0.342*"develop" + -0.316*"design" + -0.234*"research" + -0.183*"director"
Topic3 :  -0.521*"park" + -0.387*"nation" + -0.218*"langtang" + -0.171*"area" + -0.131*"altitud"
Topic4 :  0.441*"novel" + 0.245*"bunni" + 0.245*"cave" + 0.245*"nick" + 0.196*"munro"
PS C:\Users\mochhe\Documents\GradSchool\CS410\CourseProject\SourceCode>
```

11. Future Enhancements

API server

The project's backend API server is currently built to only run locally and receives HTTP GET requests on the http://localhost:4000/api endpoint. There is currently no backend API server hosted online for this project. Thus, to make this project more accessible to users, it would be greatly beneficial to have this API server hosted publicly online.

IR ALGORITHM

We plan on scaling the NLP tasks implemented in intelligentSearch.py to the bigger wiki dataset used in searchWiki.py. The Time complexity of certain NLP tasks like NER, POS tagging , applying Lemmatization and Query Expansion were high and were therefore increasing the fetch time. We saw a boost in performance with the use of above NLP techniques taking up the AP for individual queries. We therefore plan on optimizing some of the code in intelligentSearch.py to bring down the Time complexity in the future. We also plan on adding the context information from LSA into the documents and revisit our Ranking module.