PHASE ONE

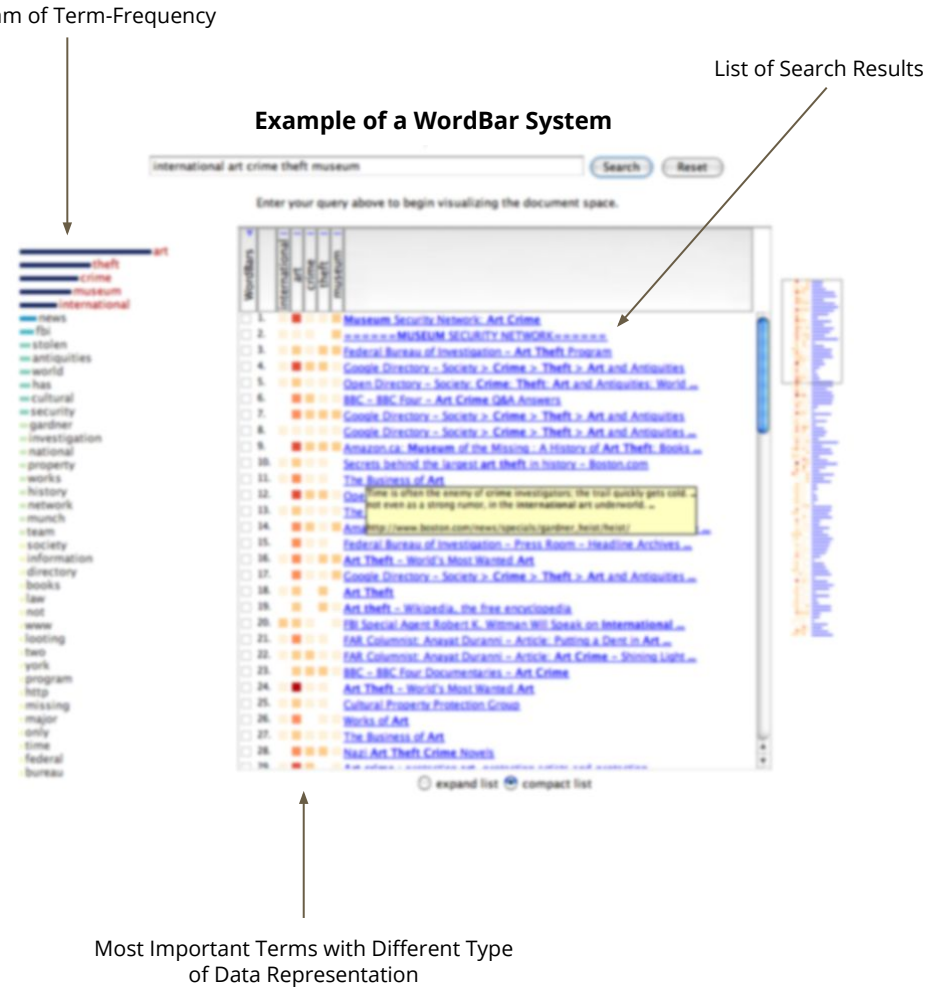
Project 15: Web search with linguistic expansions from Term-Frequency Histograms

Sean Ellis

Yu Hang Lee

Objective and Learning Goals

1. Explore how starting from a given query, alternative query formulations can be devised that can help improve information retrieval on the web
2. Involve the use of WordBars System
3. Implement the WordBars methodology
4. Identify the high frequency terms
5. Explore how well different senses of these terms can help refine the query information by using WordNet



Keywords/Phrases

WordBars

- Present the distribution of results in relation to the terms of the corresponding query and facilitate browsing of the results.
- Present with a list or bar-graph of the most frequently occurring terms in the result set of a given query

Term-Frequencies

- Represent documents as a collection of terms
- How often a term occurs in the documents

TF-IDF (Term Frequency/Inverse Document Frequency)

- weighting scheme that adjusts each frequency value by taking its product with the inverse document frequency of the term. This approach separates individual documents that may not be distinguishable from other documents that include the same term(s).

WordNet

- Online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory
- Different relations link the synonym sets

Languages/APIs: What Will Help Us Achieve Our Program Goals?

Bootstrap Framework - main source to establish the User Interface using Hypertext Markup Language/Cascading Style Sheets/JavaScript.

Python - primary back-end language for:

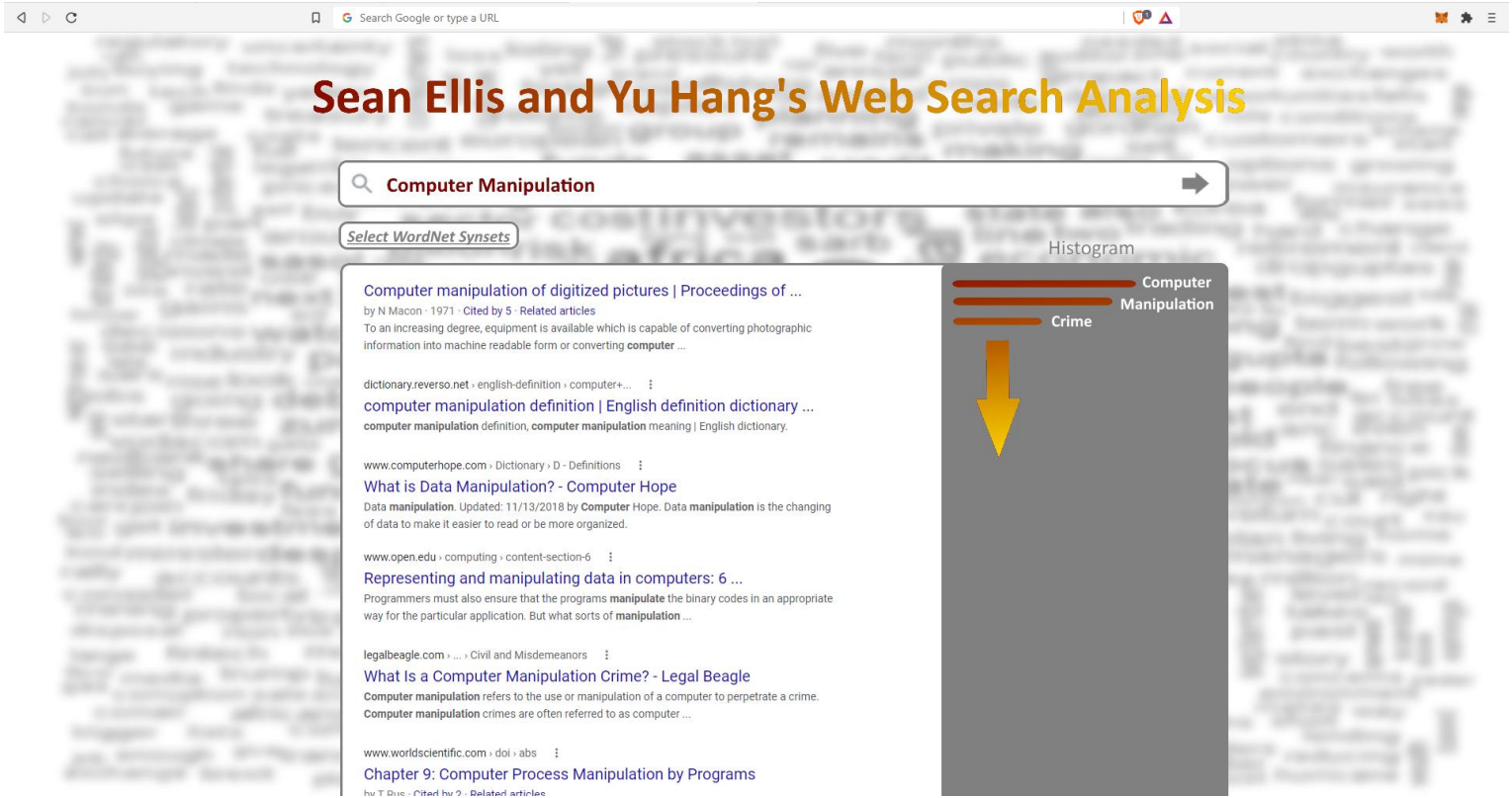
- Hosting the Web Server
- Handling all Web Server Requests made by the user
- Retrieving/Manipulating Data Queries

WordNet API - Help with refining the query to search for different relationships and words.

BeautifulSoup API - Python API that will assist in pulling Data from the found HTML Pages.

NLTK API - API that will help tokenize words, and conduct pre-filtering techniques.

Projected User Interface



Optimizations

- Implementing Singular Value Decomposition (SVD)

  Recall that the purpose of SVD is to bring forth *latent*, or hard-to-see relationships within the data.

- Pre-Filter with Stop-List/Stemming

  Note: Pre-filtering may cause slower execution and data retrieval if not implemented correctly.

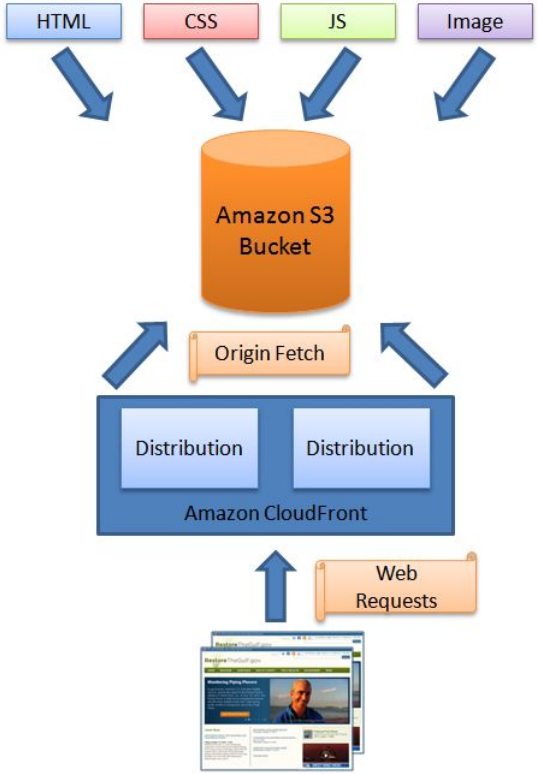
  Example of Pre-defined Stop Words with NLTK:

```
1 print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'sh
t's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves
t', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'b
ng', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'i
f', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
e', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'o
e', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'bo
me', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than',
'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 'r
n', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't
"isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "ne
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
```

Bonus Challenge

Host on EC3 Amazon



References

1. O. Hoeber and X. D. Yang, Evaluating the effectiveness of term frequency histograms for supporting interactive Web search tasks, In Proceedings of the ACM Conference on Designing Interactive Systems, pp. 360-368, 2008
2. R. Singh, Ya-Wen Hsu, and N. Moon, “Multiple-Perspective Interactive Search: A Paradigm for Exploratory Search and Information Retrieval on the Web”, Journal of Multimedia Tools and Applications, Vol. 62, pp. 507-543, 2013
3. Rahul, S. (2021). MULTIMEDIA INFORMATION SYSTEMS. *MULTIMEDIA INFORMATION SYSTEMS*, 29–80. https://ilearn.sfsu.edu/ay2021/pluginfile.php/1068666/mod_resource/content/1/MULTIMEDIA-LectureNotes-Class.pdf
4. *Create a Python Web Server - Python Tutorial*. (2020). Python. https://pythonbasics.org/webserver/
5. Otto, M. J. T. (2021). *Bootstrap*. Bootstrap. https://getbootstrap.com/

PHASE TWO

Project 15: Web search with linguistic expansions from Term-Frequency Histograms

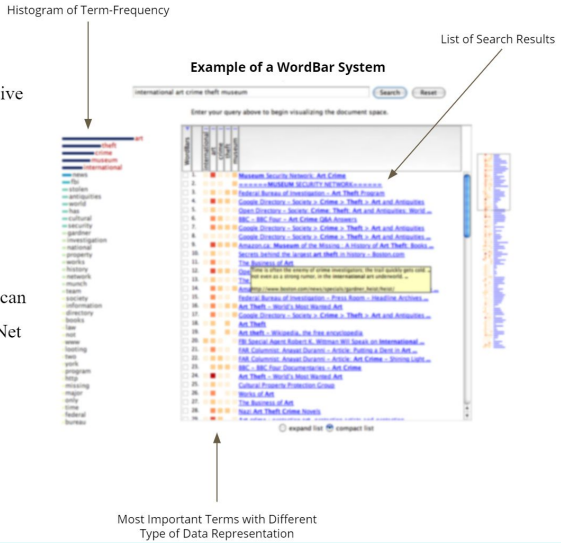
Sean Ellis

Yu Hang Lee

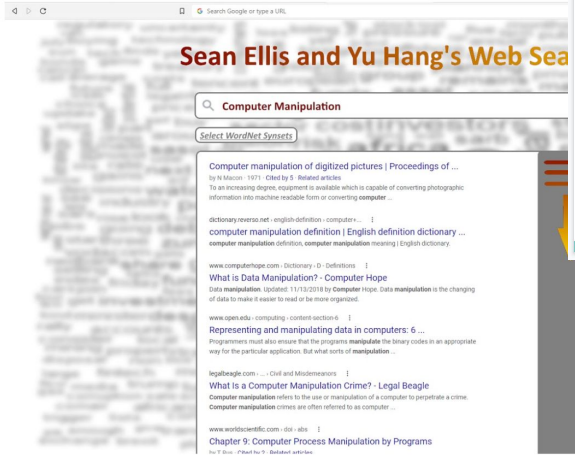
Phase One Summary

Objective and Learning Goals

1. Explore how starting from a given query, alternative query formulations can be devised that can help improve information retrieval on the web
2. Involve the use of WordBars System
3. Implement the WordBars methodology
4. Identify the high frequency terms
5. Explore how well different senses of these terms can help refine the query information by using WordNet



Projected User Interface



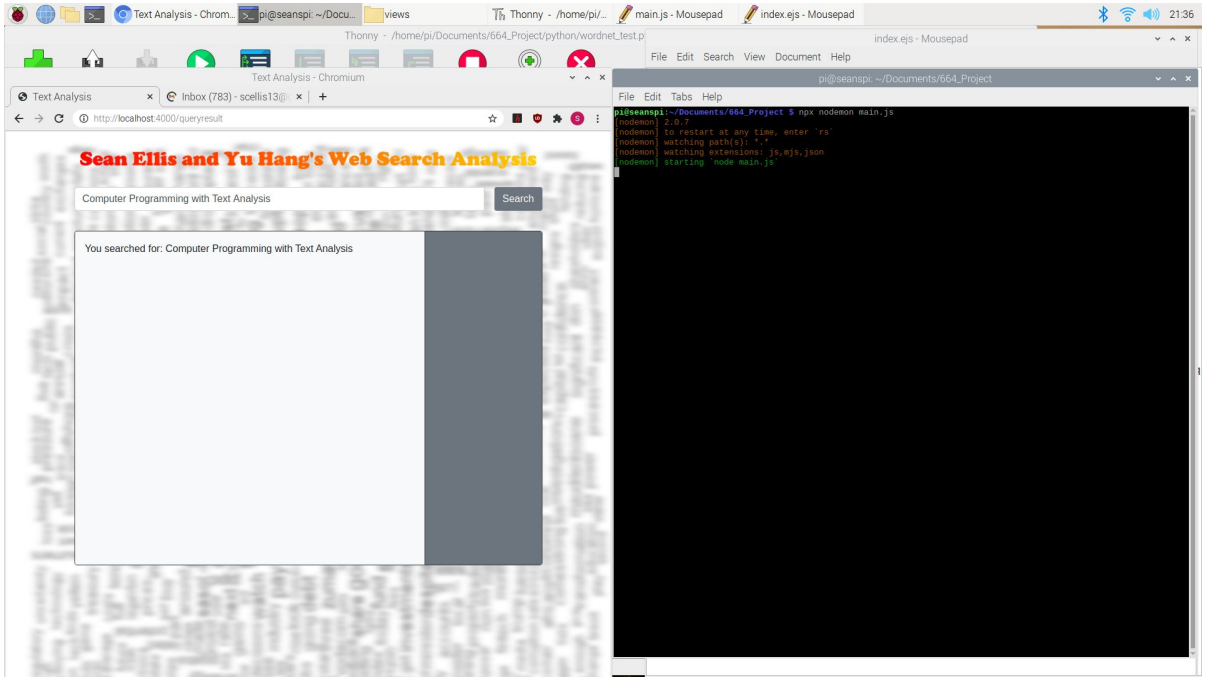
Front End Progress

Current Features:

- Using ExpressJS with EJS
- Hosting as Localhost
- Can serve data from server to client
- Can process search request.

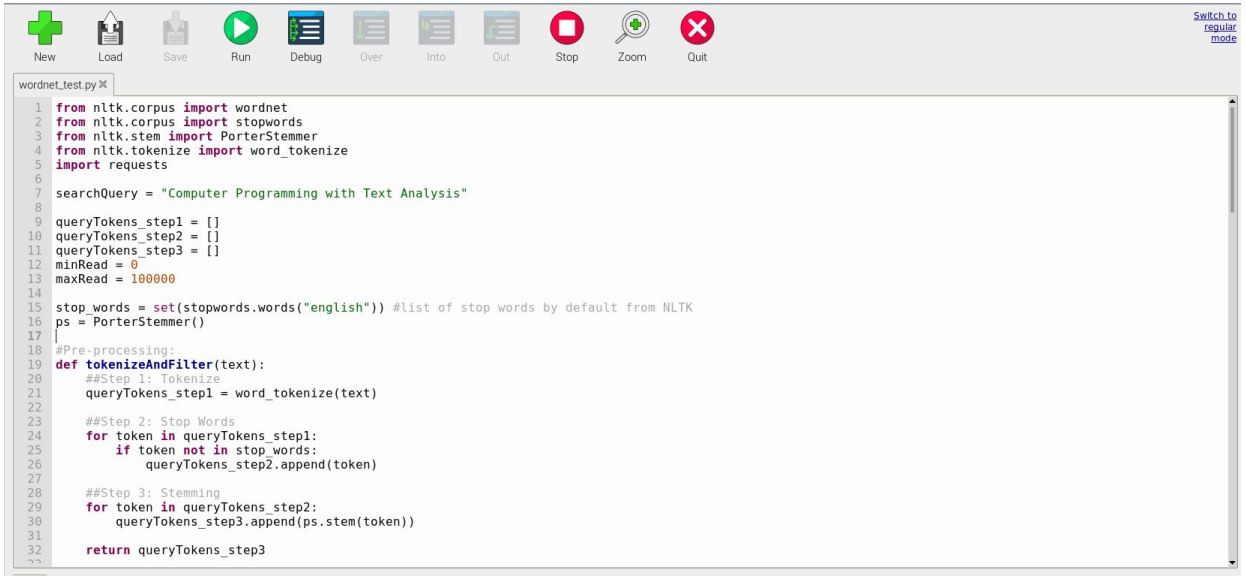
Challenges with Front-End

- Accessing Python/JSON elements
- Keeping design simple
- Connecting it to Python Scripts for Query Analysis
- ***Finding a simple API to conduct Web Searches***



Back End Progress

Beginnings of Text Analysis: Pre-processing/FIltering



Current Features:

- Using NLTK and Python Scripting
- Pre-processing given text through:
  - Tokenizing
  - Filtering (Stop-List, Stem)
- Word Counting

Challenges with Back-End

- **Properly utilizing WordNet to optimize results.**
- **Single thread use is increasing process time**
- *****Connecting Python script to the Web Server JavaScript*****

Back End Progress

Beginnings of Text Analysis: Pre-processing/FIltering - Code Continued

wordnet_test.py

```
#Pre-processing:
def tokenizeAndFilter(text):
    ##Step 1: Tokenize
    queryTokens_step1 = word_tokenize(text)

    ##Step 2: Stop Words
    for token in queryTokens_step1:
        if token not in stop_words:
            queryTokens_step2.append(token)

    ##Step 3: Stemming
    for token in queryTokens_step2:
        queryTokens_step3.append(ps.stem(token))

    return queryTokens_step3

print("First five search results with Query: ", searchQuery)
url1 = "https://monkeylearn.com/text-analysis/"
url2 = "https://matrix.berkeley.edu/research/tips-computational-text-analysis/"
url3 = "https://www.predictiveanalyticstoday.com/top-free-software-for-text-analysis-text-mining-text-analytics/"
url4 = "https://guides.temple.edu/corpusanalysis"
url5 = "https://uk.sagepub.com/en-gb/eur/computer-assisted-text-analysis/book205979"

unfilteredTokens = word_tokenize(searchQuery)

print("\n\nCounting Words from with unfiltered query: ", unfilteredTokens)

def count_tokens(textList, tokenList):
    wordFreq = []
    for word in tokenList:
        wordFreq.append(textList.count(word))
    return str(list(zip(tokenList, wordFreq)))
```

Current Features:

- Using NLTK and Python Scripting
- Pre-processing given text through:
  - Tokenizing
  - Filtering (Stop-List, Stem)
- Word Counting

Challenges with Back-End

- **Properly utilizing WordNet to optimize results.**
- **Single thread use is increasing process time**
- *****Connecting Python script to the Web Server JavaScript*****

Back End Progress

Beginnings of Text Analysis: Test 1 - Small Range of Characters

URL Not-Filtered
Query Not-Filtered

URL Not-Filtered
Query Filtered

URL Filtered
Query Filtered

```
Python 3.7.3 (/usr/bin/python3)
>>> %Run wordnet_test.py

  First five search results with Query:  Computer Programming with Text Analysis


  Counting Words from with unfiltered query:  ['Computer', 'Programming', 'with', 'Text', 'Analysis']
           https://monkeylearn.com/text-analysis/
           [('Computer', 0), ('Programming', 0), ('with', 0), ('Text', 0), ('Analysis', 0)]
           https://matrix.berkeley.edu/research/tips-computational-text-analysis/
           [('Computer', 0), ('Programming', 0), ('with', 0), ('Text', 1), ('Analysis', 1)]
           https://www.predictiveanalyticstoday.com/top-free-software-for-text-analysis-text-mining-text-analytics/
           [('Computer', 0), ('Programming', 0), ('with', 0), ('Text', 0), ('Analysis', 0)]
           https://guides.temple.edu/corpusanalysis
           [('Computer', 0), ('Programming', 0), ('with', 0), ('Text', 0), ('Analysis', 3)]
           https://uk.sagepub.com/en-gb/eur/computer-assisted-text-analysis/book205979
           [('Computer', 0), ('Programming', 0), ('with', 0), ('Text', 0), ('Analysis', 0)]


  Counting Words from with filtered tokens:  ['comput', 'program', 'text', 'analysi']
           https://monkeylearn.com/text-analysis/
           [('comput', 0), ('program', 0), ('text', 0), ('analysi', 0)]
           https://matrix.berkeley.edu/research/tips-computational-text-analysis/
           [('comput', 0), ('program', 0), ('text', 0), ('analysi', 0)]
           https://www.predictiveanalyticstoday.com/top-free-software-for-text-analysis-text-mining-text-analytics/
           [('comput', 0), ('program', 0), ('text', 0), ('analysi', 0)]
           https://guides.temple.edu/corpusanalysis
           [('comput', 0), ('program', 0), ('text', 0), ('analysi', 0)]
           https://uk.sagepub.com/en-gb/eur/computer-assisted-text-analysis/book205979
           [('comput', 0), ('program', 0), ('text', 0), ('analysi', 0)]

  Counting Words with filtered tokens AND filtered text tokens:  ['comput', 'program', 'text', 'analysi']
           https://monkeylearn.com/text-analysis/
           [('comput', 2), ('program', 2), ('text', 2), ('analysi', 2), ('comput' …
           https://matrix.berkeley.edu/research/tips-computational-text-analysis/
           [('comput', 4), ('program', 3), ('text', 4), ('analysi', 4), ('comput' …
           https://www.predictiveanalyticstoday.com/top-free-software-for-text-analysis-text-mining-text-analytics/
           [('comput', 6), ('program', 4), ('text', 6), ('analysi', 6), ('comput' …
           https://guides.temple.edu/corpusanalysis
           [('comput', 11), ('program', 5), ('text', 8), ('analysi', 11), ('compu …
           https://uk.sagepub.com/en-gb/eur/computer-assisted-text-analysis/book205979
           [('comput', 16), ('program', 6), ('text', 10), ('analysi', 16), ('comp …
>>>
```

Note: URL's are first five Google Search Results when using "Computer Programming with Text Analysis"

Back End Progress

Beginnings of Text Analysis: Test 2 - Large Range of Characters

Test Difference: More words to compare at the start of the HTML Document.

URL Not-Filtered
Query Not-Filtered

URL Not-Filtered
Query Filtered

URL Filtered
Query Filtered

Shell

```
Python 3.7.3 (/usr/bin/python3)
>>> %Run wordnet_test.py
  First five search results with Query:  Computer Programming with Text Analysis


  Counting Words from with unfiltered query:  ['Computer', 'Programming', 'with', 'Text', 'Analysis']
           https://monkeylearn.com/text-analysis/
           [('Computer', 0), ('Programming', 0), ('with', 0), ('Text', 0), ('Analysis', 0)]
           https://matrix.berkeley.edu/research/tips-computational-text-analysis/
           [('Computer', 1), ('Programming', 0), ('with', 9), ('Text', 14), ('Analysis', 13)]
           https://www.predictiveanalyticstoday.com/top-free-software-for-text-analysis-text-mining-text-analytics/
           [('Computer', 0), ('Programming', 0), ('with', 0), ('Text', 0), ('Analysis', 0)]
           https://guides.temple.edu/corpusanalysis
           [('Computer', 0), ('Programming', 0), ('with', 9), ('Text', 1), ('Analysis', 9)]
           https://uk.sagepub.com/en-gb/eur/computer-assisted-text-analysis/book205979
           [('Computer', 4), ('Programming', 0), ('with', 11), ('Text', 11), ('Analysis', 18)]


  Counting Words from with filtered tokens:  ['comput', 'program', 'text', 'analysi']
           https://monkeylearn.com/text-analysis/
           [('comput', 0), ('program', 0), ('text', 6), ('analysi', 0)]
           https://matrix.berkeley.edu/research/tips-computational-text-analysis/
           [('comput', 0), ('program', 3), ('text', 34), ('analysi', 0)]
           https://www.predictiveanalyticstoday.com/top-free-software-for-text-analysis-text-mining-text-analytics/
           [('comput', 0), ('program', 0), ('text', 0), ('analysi', 0)]
           https://guides.temple.edu/corpusanalysis
           [('comput', 0), ('program', 0), ('text', 14), ('analysi', 0)]
           https://uk.sagepub.com/en-gb/eur/computer-assisted-text-analysis/book205979
           [('comput', 0), ('program', 0), ('text', 7), ('analysi', 0)]


  Counting Words with filtered tokens AND filtered text tokens:  ['comput', 'program', 'text', 'analysi']
           https://monkeylearn.com/text-analysis/
           [('comput', 2), ('program', 2), ('text', 8), ('analysi', 2), ('comput' …
           https://matrix.berkeley.edu/research/tips-computational-text-analysis/
           [('comput', 46), ('program', 19), ('text', 79), ('analysi', 40), ('com …
           https://www.predictiveanalyticstoday.com/top-free-software-for-text-analysis-text-mining-text-analytics/
           [('comput', 90), ('program', 36), ('text', 150), ('analysi', 78), ('co …
           https://guides.temple.edu/corpusanalysis
           [('comput', 154), ('program', 54), ('text', 255), ('analysi', 139), (' …
           https://uk.sagepub.com/en-gb/eur/computer-assisted-text-analysis/book205979
           [('comput', 224), ('program', 73), ('text', 378), ('analysi', 224), (' …
>>>

Python 3.7.3 (/usr/bin/python3)
>>>
```

URL Result #5 matches best so far by Word Count/Filtering alone.

New Challenges Arise

1. How are we going to efficiently tokenize and filter, and return results back to the user in a *reasonable time*?
2. How are we going to connect our Python data to our Web Server?
3. **How can we utilize WordNet and Synsets to optimize the results?**
4. What better API's/Languages can we code with/instead to be more productive in Text Analysis?
5. How can we use NLTK's Word Similarities Percentages, and use it to our advantage when comparing results, or tokens?

References

1. O. Hoeber and X. D. Yang, Evaluating the effectiveness of term frequency histograms for supporting interactive Web search tasks, In Proceedings of the ACM Conference on Designing Interactive Systems, pp. 360-368, 2008
2. NLTK Toolkit. *Natural Language Toolkit.* https://www.nltk.org/. Accessed on 4/1/2021.

Project Contributions

Sean Ellis

Web Server Hosting, NLTK Implementation/Importing, Python Scripting, ExpressJS/EJS, Slideshow Organization

Yu Hang Lee

WordNet Application, Python Scripting, Slideshow Organization, Project Overview

Final Phase

Project 15: Web search with linguistic expansions from Term-Frequency Histograms

Sean Ellis

Yu Hang Lee

End Goal In Sight?

Initial Discoveries:

1. Calculating query refinement comes at a major sacrifice to time/performance, if performed in real-time.

2. *Figuring out the right combination of Synset values can be complex.

3. Performing mathematical calculations without libraries is troublesome.

Major Question

The assignment focuses on identifying high frequency terms, and to explore how well different senses of these terms can help refine the query formulation . But the questions still arise...

Potential Case Studies

Compare/Analyze the Results

$$w_{i,j} = tf_{i,j} \times \log_2\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

How We Will Handle Including Tokens to Compare Text

How We Will Handle the Web Engine Search

Case 1: Original Query Original Tokens	Case 2: Original Query Synset Tokens	Case 3: Original Query Combine Tokens
Case 4: New Query Original Tokens	Case 5: New Query Synset Tokens	Case 6: New Query Combine Tokens
Case 7: Combine Query Original Tokens	Case 8: Combine Query Synset Tokens	Case 9: Combine Query Combine Tokens

References

1. O. Hoeber and X. D. Yang, Evaluating the effectiveness of term frequency histograms for supporting interactive Web search tasks, In Proceedings of the ACM Conference on Designing Interactive Systems, pp. 360-368, 2008
2. NLTK Toolkit. *Natural Language Toolkit.* https://www.nltk.org/. Accessed on 4/1/2021.
3. O. Hoeber and X. D. Yang, Evaluating the effectiveness of term frequency histograms for supporting interactive Web search tasks, In Proceedings of the ACM Conference on Designing Interactive Systems, pp. 360-368, 2008
4. R. Singh, Ya-Wen Hsu, and N. Moon, "Multiple-Perspective Interactive Search: A Paradigm for Exploratory Search and Information Retrieval on the Web", Journal of Multimedia Tools and Applications, Vol. 62, pp. 507-543, 2013
5. Rahul, S. (2021). MULTIMEDIA INFORMATION SYSTEMS. *MULTIMEDIA INFORMATION SYSTEMS*, 29–80. https://ilearn.sfsu.edu/ay2021/pluginfile.php/1068666/mod_resource/content/1/MULTIMEDIA-LectureNotes-Class.pdf
6. *Create a Python Web Server - Python Tutorial*. (2020). Python. https://pythonbasics.org/webserver/
7. Otto, M. J. T. (2021). *Bootstrap*. Bootstrap. https://getbootstrap.com/

Project Contributions

Sean Ellis

Front/Back End Application; Python, ExpressJS, EJS Development; Slideshow Creation, Project Research

Yu Hang Lee

Project Research, Report Research, Wordnet Research