Squad

Team members:

Kurt Bognar, Zach Bryant, Ali Raza, Justin Renneke, Jared Welch

Project:

Disclosure

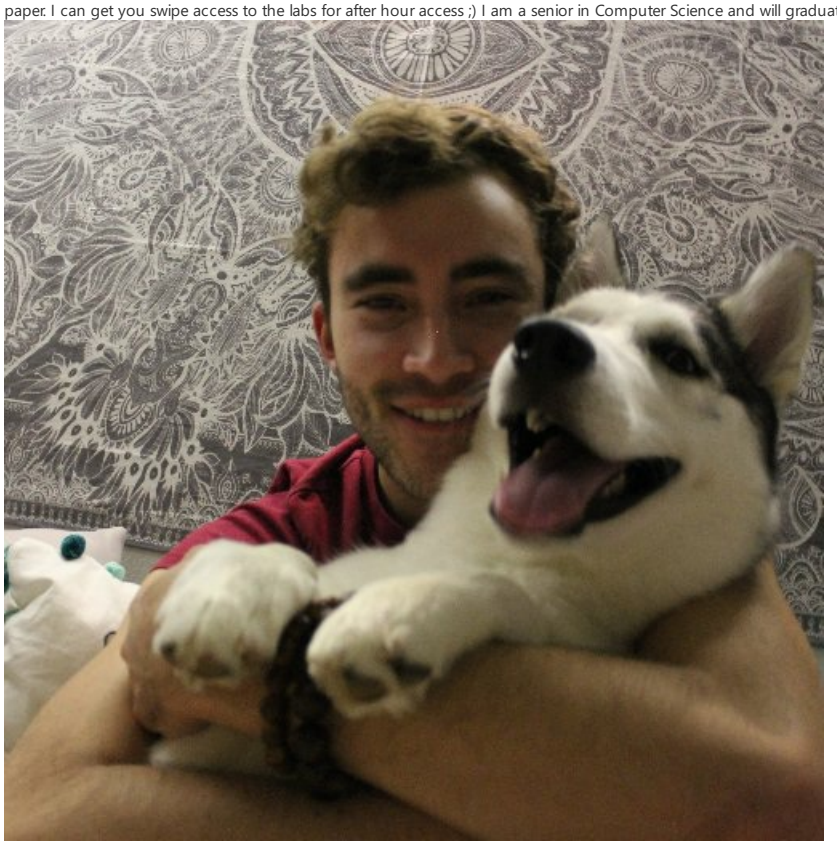
Team Information

Team Name: Squad

Mission Statement: Our mission is to gather news articles and then categorizes them based upon similarity after processing our data with Natural Language Processing techniques. We will provide visualizations of our results to user via a web application. By doing this we hope to expose trends in the news.

Team Members:

- **Zach Bryant:** Nickname = Zen Master Zach. I work on campus as an IT Support Specialist. Yes, I save lives. Really though, I am in charge of keeping all the printers in the computer labs filled with paper. I can get you swipe access to the labs for after hour access ;) I am a senior in Computer Science and will graduate December 2017.



- **Kurt Bognar:** Senior at MU Majoring in Computer Science with a Minor in Math. I'm a research for the university aiding in the creation of new tools and algorithms for Association-Rule Mining.

- **Ali Raza:** I am a senior at the University of Missouri studying Computer Science. I do research at the iDAS Lab where I am currently developing a data mining library.

- **Jared Welch:** Senior at the University of Missouri. Passionate about improving my skills. Excited for the potential of our application.

- **Justin Renneke:** Senior undergraduate of Computer Science at the University of Missouri, graduating December '17. Interested in machine learning, data analysis, cloud computing, and using Python to conquer the galaxy.

Introduction

Problem Definition

News articles generate massive amounts of data. Following the climactic nature of the 2016 Presidential Election due to the dissemination of false news, it has become evident that it is important to analyze news articles. However, up until recent years, it has not been possible to effectively analyze this data due to computational limitations. However, recent advances in High Performance Computing, Data Analytics, and Machine Learning have made it possible to gather, store, and analyze news articles. We propose a model to cluster news articles based upon similarity and then provide data visualizations based upon the generated clusters and analysis done.

Problem Resolution

We propose the following software solution:

- Scrape news articles.
- Pre-process using NLP and then perform TFIDF.
- Cluster news articles based on the similarity of TFIDF vectors.
- Display visualizations

Changelog for software

NewsArticleScraper - Version 1.1

- Initial features:
  - automatically scheduled at time during the night
  - each site runs as an independent process, so any error for a site does not hurt the flow of the scraper itself
  - as articles are scraped from a site, they are pushed into the database, including the text, title, authors, url, and other relevant info
- Version 1.1:
  - add additional sites to list
  - clean up inserts to database
  - add custom logging functionality, including error and information log files
  - more try catch error checking to improve error logs

Requirements

User Requirements

- User can explore visualizations of statistics and analysis of news data using a graphical interface to navigate.
  - Users can use the website to view clusters of articles by topic via a visualization.
  - Users can use the website to view visualizations depicting trending topics per day for a past time period.
  - Users can use the website to view a word cloud representation of trending topics for that day.
- User can submit an article to be analyzed and view the results of the analysis of the article compared to our dataset; the results will be classification data based upon our models.
- **User Requirements Stretch Goals:**
  - User can perform custom searches against the database, yielding search results related to statistics on the news article data
  - User can explore articles pertinent to trending twitter topics.

Functional Requirements

- Given a list of news websites, scrape every new article on every site and return and store in a database the following data from each article: Article title, author name(s), date published, article body text, raw html, and webpage url.
- Perform natural language processing analysis on articles to clean the data and generate features such as named entities, bag of word counts, and term frequency-inverse document frequency metrics.
- Categorize articles into groups based on topic determined by performing machine learning-based cluster analysis using generated features.
- A web application will provide users with visualizations of the clustered topics and their articles.
- The web application will allow a user to provide the url of a specific news article, scrape the webpage, assign the article to a topic category, and inform the user of the result.
- **Functional Requirements Stretch Goals**
  - Allow users to perform custom searches of the article database and return data visualizations based on metrics such as frequency of occurrence, trends of occurrence over time, and relation to other topics.
  - Create a scrolling wordcloud graphic to visualize changes in overarching news trends over time.
  - Perform sentiment analysis on articles and visualize the results.

Non Functional Requirements

- The server is expected to have a 99% availability.
- Each layer of the system should be backed-up weekly
- The web application will have HTTPs encryption with a valid SSL certificate.
- The software will be written using modular OOP practices to allow for easy addition of new features.
- The web application will be designed to be compatible on mobile and tablet devices.
- The system will decouple an RDS cloud database from computational resources to enable vertical scaling as needed.
- **Non Functional Requirements Stretch Goal**
  - Response Time: Real time article classification

System Requirements

- The project will be hosted on decoupled Amazon AWS instances.
  - A web scraper, database system, and web application will each have their own isolated instance.
  - The system will auto scale for intensive data analysis and improved response time during high usage situations.
- A relational database will be used in order to be able to accurately model the data and perform advanced queries.
  - The database should be large enough to store a massive dataset.
  - The database should be indexed so efficient queries can be performed.
- The computational processing power must be sufficient enough to not bottleneck the clustering, analysis, and data visualization process.
- The Random-Access-Memory must be high enough to minimize reading from disk during computationally intensive tasks.
- **System Requirements Stretch Goals:**
  - Store articles using a distributed framework such as Hadoop.
  - Implement a cluster computing system and perform parallel analysis in order to drastically improve computational speed.

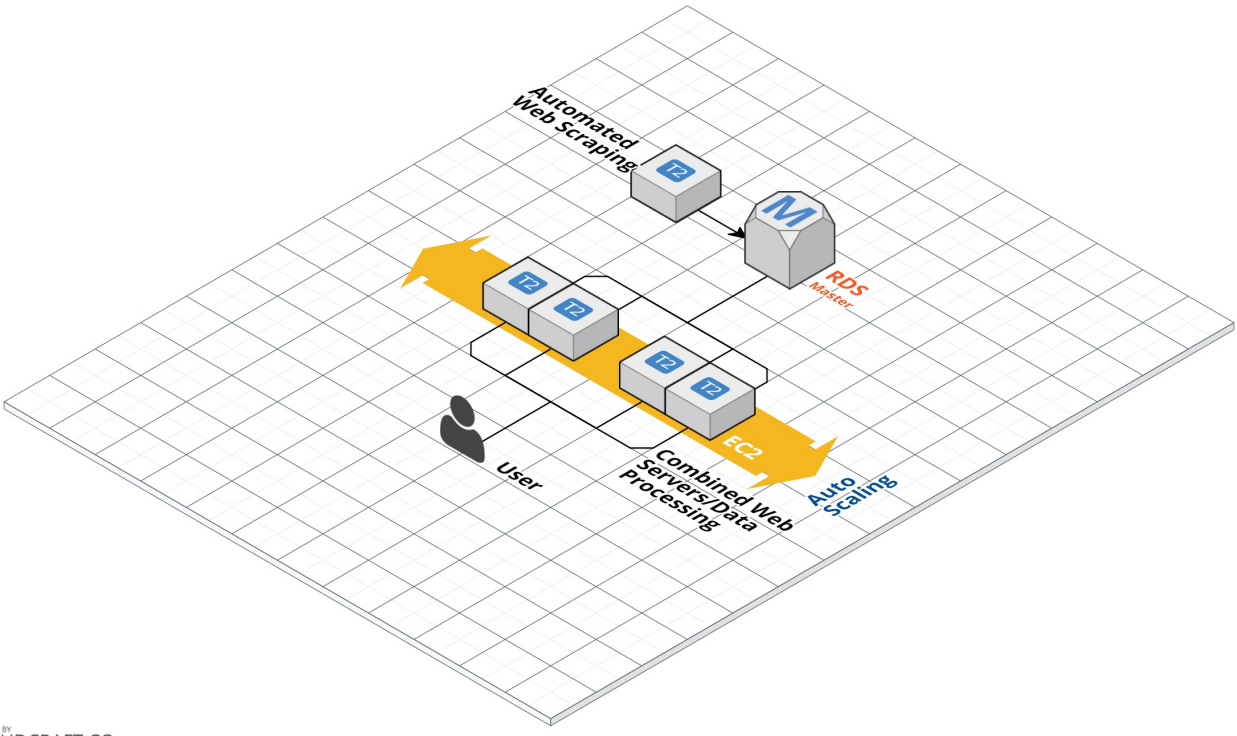
Design

Phase I: System Design

The project will be deployed within Amazon Web Services. There are two architecture proposals: one for the prototyping phase (which will be the entire extent of the capstone project) and one for a business model deployment (in case the project is to deploy for real world use).

Decoupling and auto scaling lie at the heart of each design. A decoupled web scraping instance will allow the web scraper to continue to run uninterrupted in an automated environment. Auto Scaling groups of larger EC2 instances will allow the design to scale up for processing intensive data analysis and faster response to user queries when under load, while conserving resources by scaling down when the system is idle. To achieve this, the database must be decoupled as well.

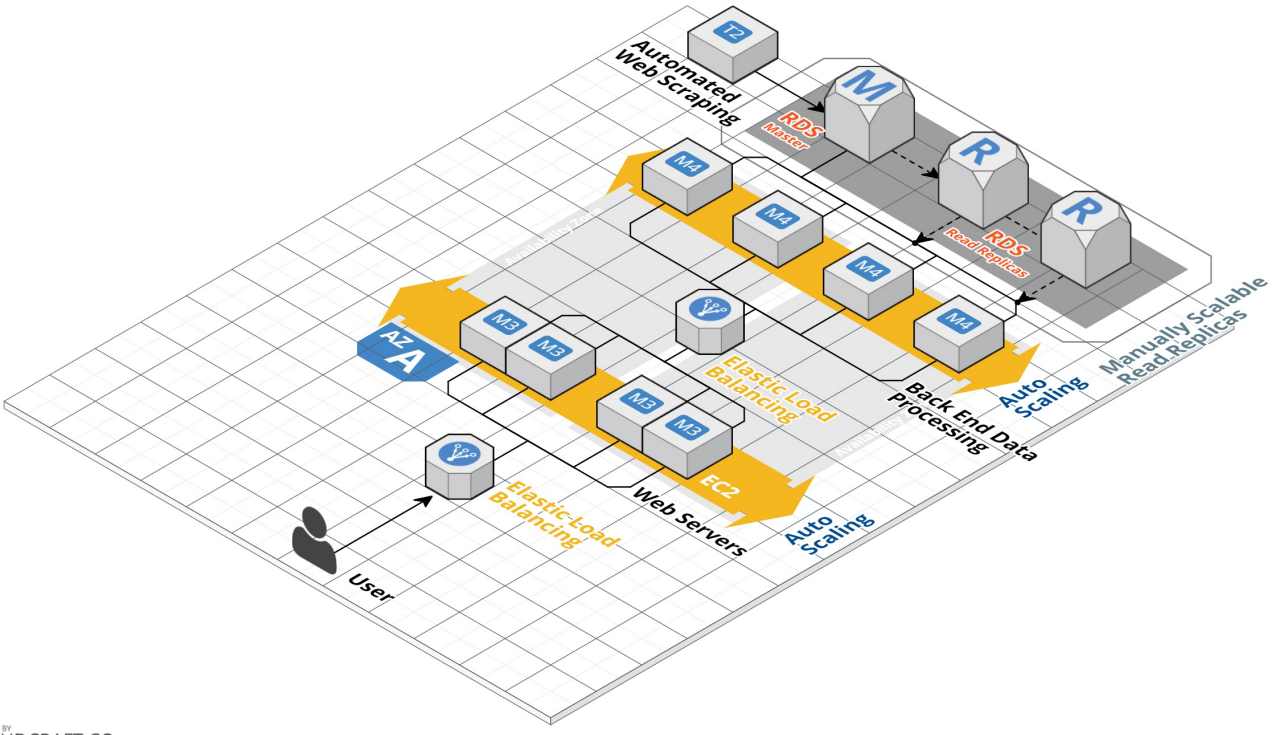
AWS Prototyping Deployment



Features:

- Auto Scaling combined Web Servers/Data Processing
  - This provides scale-up capability for data processing and to process individual user queries in a prototyping environment
  - Saves money by reducing resource consumption since the system will not need to handle many users in a prototyping environment and the system can afford to sacrifice some responsiveness for the sake of cost
- Single, decoupled RDS database
- Decoupled web scraper

AWS Business Model Deployment

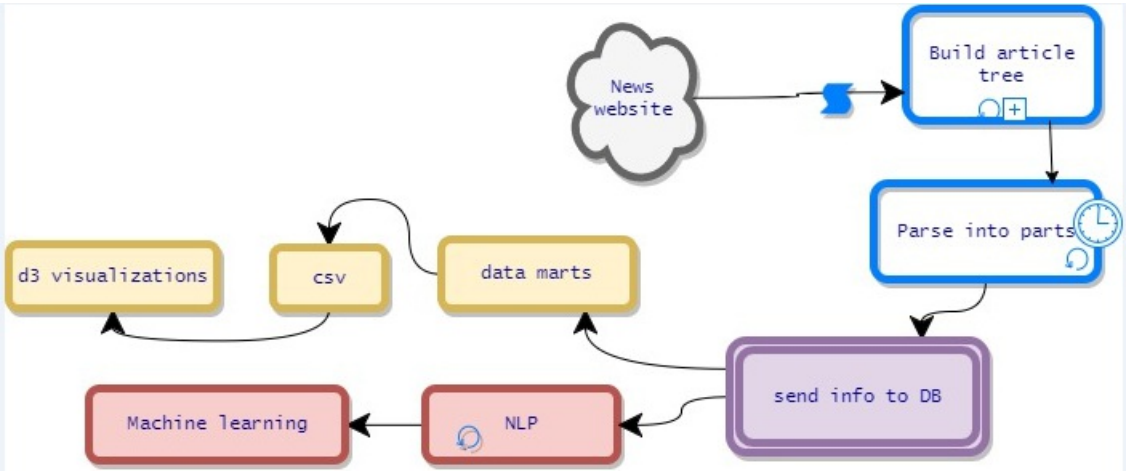


POWERED BY CLOUDCRAFT.CO

Features:

- Elastic load balancing will redirect work and balance load at two levels: User access of web servers on the front end and data processing and data queries on the back end.
- Multiple auto scaling groups will provide high performance and redundancy across AWS availability zones for a larger business environment deployment
  - SSD-based M3 web server auto scaling group will provide a high performance cluster for front-end processing
  - AWS Elastic block store-based M4 data processing auto scaling group will provide high processing capacity for data analysis and queries
  - Decoupled RDS database scaling group with a Master RDS database feeding RDS Read Replicas will allow us to easily scale manually to provide high database throughput for intense data queries
- Decoupled web scraper will continue to run its relatively undemanding task undisturbed

High Level Logical System Flow



Technologies Used

This is a list of the technologies that will be used for reference. These are discussed in more detail at various places throughout this document.

- **Amazon Web Services**
- **PostgreSQL**
- **Python Back-End**
  - **Important libraries:**
    - Web scraping: Newspaper
    - Natural language processing: NLTK and Stanford OpenNLP
    - Machine learning: skLearn
    - Data manipulation: Pandas
  - **Flask (web framework)**

- **HTML/Javascript Front-End**
    - **D3.js library for visualizations**

Phase II: Web Scraping and Data Design

This project will require a large database of news articles and their associated metadata. To this purpose a web scraper has been built to scrape articles from news sites and a data warehouse to store the articles and their metadata. As of the time of writing, over 30,000 articles have been collected and there are plans to run the scraper through the summer to accumulate a few hundred thousand articles for data analysis in Capstone II.

Web scraper

Overview

- Parses a list of multiple news sites. This list is provided by a JSON file containing entries with a site name and URL.
- Carefully designed to avoid website's banning our IP
    - The parser is run automatically every day at a random time between 2 AM and 6 AM by using the apscheduler Python library.
    - A random wait is added between page scrapes
    - Runs with threading on each site, but limited to 2 threads to avoid stressing site's resources
- Implements logging to help with troubleshooting an otherwise fully automated program
- Every individual site crawl and scrape is forked out as a separate process to improve performance and avoid memory issues

Python's Newspaper Library

Link to Newspaper library Homepage

The Newspaper library is specifically designed to scrape articles from news websites. Newspaper itself is based on Python's popular BeautifulSoup and Goose parsing libraries. This library provides the ability to cleanly scrape news articles from a wide variety of websites - a challenge that would otherwise be incredibly time consuming. The library accomplishes this by: querying a website's home page, crawling through all the links associated with the website, building a tree structure to represent it, then scraping and parsing every previously unseen article in the tree.

Using this library, it is possible to collect the following data from every article:

- Title
- Primary author (if exists)
- Secondary authors (if exists)
- Date published (if exists)
- URL
- Body text
- Raw html

The library also has a memoization capability that is being used. This allows the scraper to scrape only new articles for every subsequent scrape of a site.

Scraper Code/Pseudocode

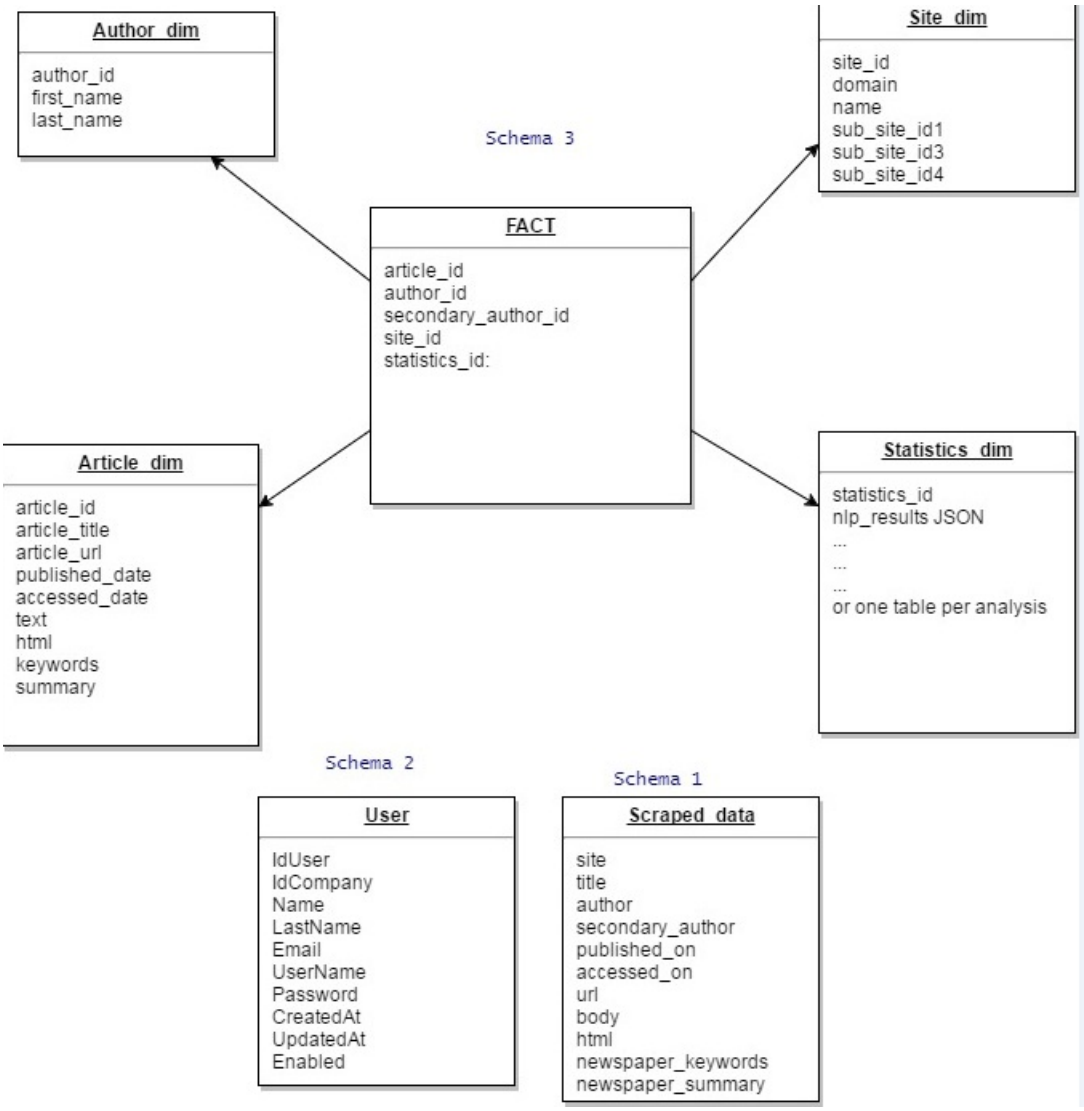
```
# Open database connection
# Read in site list JSON

# Enter core scraping code
for site in site_list:
                child_id = os.fork()

                if child_id == 0:
                        name = site['name']
                        url = site['url']
                        # if a site should throw an exception for any reason (maybe bad url), catch it, pass, and move to next site
                        try:
                                paper = newspaper.build(url,
                                                keep_article_html=True,
                                                fetch_images=False,
                                                memoize_articles=True,  # track the articles we have already scraped from session to session
                                                MIN_WORD_COUNT=200,             # this doesn't seem to be having any affect
                                                number_threads=2,
                                                request_timeout=12,
                                                thread_timeout=3)
                                total = paper.size()
                                numScraped = total
                                authors = []
                                secondAuth = ''

                                for x in range(0,numScraped):
                                        sleep(randint(3,6))
                                        if paper.articles[x] != None:
                                                paper.articles[x].download()
                                                if paper.articles[x].is_downloaded:
                                                        if paper.articles[x].html != None:
                                                                paper.articles[x].parse()
                                                                # Ensure article is long enough to be valid
                                                                if len(nltk.word_tokenize(paper.articles[x].text)) > 200:
                                                                        html = paper.articles[x].article_html.replace('\n', ' ')
                                                                        title = (paper.articles[x].title)
                                                                        url = paper.articles[x].url
                                                                        published_date = paper.articles[x].publish_date
                                                                        authors = paper.articles[x].authors
                                                                        text = paper.articles[x].text.replace('\n', ' ')
                        except Exception as e:
                                logging.error("::Exception:Scrapping error for site: " + name + "::")
                                logging.error("::Exception: " + str(e) + "::")
                                pass
                        os._exit(0)
# Clean author strings
# Insert into database
# Log any other errors
```

Data Design



- ERD Diagram
  The database has 3 schemas. The reason for this is that there are three different access points. There are reporting mechanisms that would use the data warehouse in schema 3. In schema 2, the user information for accessing the website will exist. In schema 1 only the data warehouse and the scraper will have access to this transactional table which is a flow of articles. The reason for creating the data warehouse is because it allows you to create data marts that will join together all of the information you use easier for reporting. Most likely each visualization will have a data mart.
- RDBMS: PostgreSQL is the database system to be used for a variety of reasons:
  - It is a relational database which is needed because the data being gathered by our scraper has many complex relationships.
  - PostgreSQL supports advanced data-types such as JSON while also providing object features which will allow for easy object-relational mapping if needed.
- ETL Pipeline:
  - Extraction: The articles are extracted by a webscraper. The scraper places all the data collected into a single database table.
  - Transformation: The data is organized into relational tables through SQL queries.
  - Loading: The relational tables will be used in conjunction with machine-learning analysis to gather meaningful and interesting information.

Phase III: Data Analysis

Pre-Processing and Feature Extraction

- **Natural Language Processing**

  Various NLP methods will play a key role in cleaning our data and extracting features for use in machine learning analysis. Some of these methods do similar things in different ways and the viability of the various methods will need to be explored as the data is explored.

  - **Removal of Stopwords**

    Stopwords are the very common words in the English language such as 'the', 'there', 'from', etc that provide little to no information on their own. The project will use NLP to remove these words before performing further analysis. Python's NLTK library offers predefined lists of stopwords and functions to easily accomplish this.

    Example code:

    ```
    from nltk.corpus import stopwords
    stopwords.words('english')
    # For every article in the database, filter out the stopwords and return the filtered text using a list comprehension
    for article in article_database:
            filtered_article = [word for word in word_list if word not in stopwords.words('english')]
    ```

  - **Stemming**

    Stemming is the process of reducing topically similar words to their roots. For example, “stemming,” “stemmer,” “stemmed,” all have similar meanings; stemming reduces those terms to “stem.” This is an important feature for understanding the nature of a text's topic, which would otherwise view those terms as separate entities and reduce their importance in the model. The NLTK Python library offers a stemmer function based on the most widely used stemming algorithm: Porter's stemmer.

```
from nltk.stem.porter import PorterStemmer

# Create p_stemmer of class PorterStemmer
p_stemmer = PorterStemmer()

# stem tokens by applying the stemmer to every token in a list of tokens, returning a list of stems
texts = [p_stemmer.stem(i) for i in tokens_list]
```

- **Lemmatization**

  Lemmatization is similar in concept to stemming, but there are important differences. Both are an attempt to find the root of a word from amongst its many forms to reduce data dimensionality, but, where stemming uses a relatively unsubtle chopping heuristic to chop word endings, lemmatization uses a more comprehensive approach that takes into account the parts of speech surrounding the word in question. For example: both stemming and lemmatization would derive 'stem' from 'stemming,' 'stemmer,' and 'stemmed'; however, only lemmatization would derive 'be' from 'am', 'are', and 'is.' This increases accuracy but at the price of performance. NLTK offers lemmatization via the WordNetLemmatizer.

```
from nltk.stem import WordNetLemmatizer
sent = "cats running ran cactus cactuses cacti community communities"

wnl = WordNetLemmatizer()
" ".join([wnl.lemmatize(i) for i in sent.split()])
# outputs 'cat running ran cactus cactus cactus community community'
```

- **Named Entity Extraction**

  Named entity extraction attempts to find all named entities within a text document and categorize them as a person, location, organization, etc. The Stanford NLP library is recognized as being the best at this and Python offers access to this Java library with a wrapper. Example:

```
from nltk.tag import StanfordNERTagger
from nltk.tokenize import word_tokenize

text = "The President of the United States is named Donald Trump and he has several children including Donald Trump Jr., Invanka Trump, and a son-in-law: Jared
Kushner. He also has a wife named Melania. President Trump is the defacto head of the Republican Party, even though he identified as a Democrat for most of his
life. The Republicans have been slow to embrace Mr. Trump."

# Tokenize the text
tokenized_text = word_tokenize(text)
# Run the text through the tagger
categorized_text = st.tag(tokenized_text)
# This will return labeled tuples and, after further manipulation, we will get a list of tagged entity tuples:
[('United States', 'LOCATION'), ('Donald Trump', 'PERSON'), ('Donald Trump Jr.', 'PERSON'), ('Invanka Trump', 'PERSON'), ('Jared Kushner', 'PERSON'), ('Trump',
'PERSON'), ('Melania', 'PERSON'), ('Trump', 'PERSON'), ('Republican Party', 'ORGANIZATION'), ('Trump', 'PERSON')]
```

- **Keyword Extraction**

  Keywords attempt to describe the main topics expressed in an article. Python offers an easy to use keyword extraction library called RAKE.

  This link offers a tutorial on the RAKE workflow.

- **Frequency Distribution**

  A frequency distribution counts the number of times every word appears in a text. NLTK offers this functionality with the FreqDist() function.

```
import nltk

sentence = 'How much wood would a wood chuck chuck if a wood chuck could chuck wood?'

tokens = nltk.word_tokenize(sentence)
fdist = nltk.FreqDist(w.lower() for w in tokens)

# returns: 'wood': 4, 'chuck': 4, 'a': 2, 'would': 1, 'could': 1, 'how': 1, 'much': 1, 'if': 1
```

- **Term Frequency - Inverse Document Frequency (TF-IDF)**

  TF-IDF is a way to score the importance of words (or "terms") in a document based on how frequently they appear across multiple documents. If a word appears frequently in a document, it's important. Give the word a high score. But if a word appears in many documents, it's not a unique identifier. Give the word a low score. Python's scikit learn toolkit provides the skleam library with this functionality.

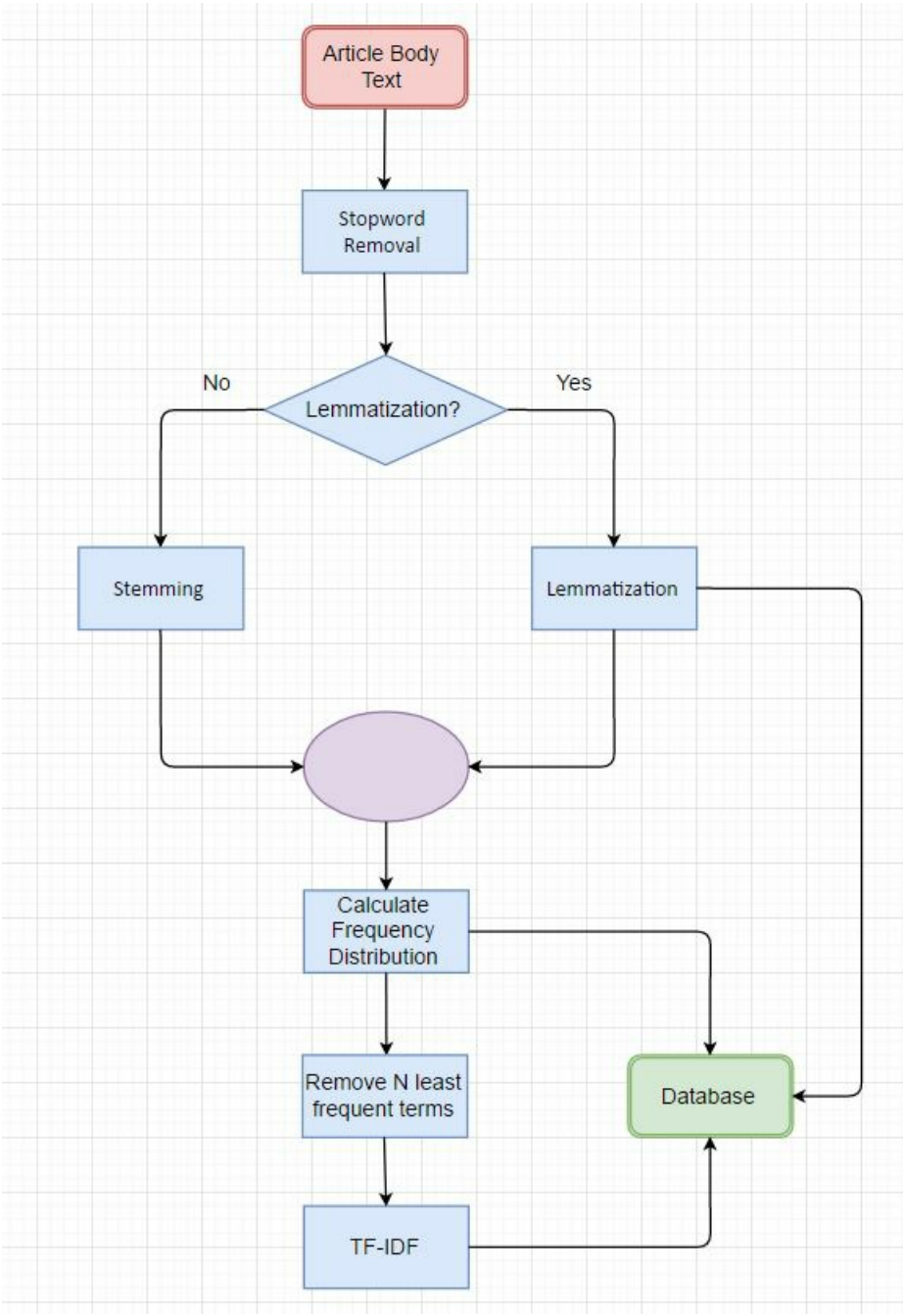
```
from sklearn.feature_extraction.text import TfidfVectorizer

# create a TfidfVectorizer object
tfidf = TfidfVectorizer(tokenizer=tokenize, stop_words='english')
# give the vectorizor a dictionary of documents containing document:content pairs to vectorize
tfs = tfidf.fit_transform(token_dict.values())
```

  The code snippet seen above, if run on a corpus containing the works of Shakespeare, will return a matrix containing the following values for the selected words:
  unseen - 0.309281094362
  lord - 0.156737043549
  king - 0.164996828044
  juliet - 0.544613034225

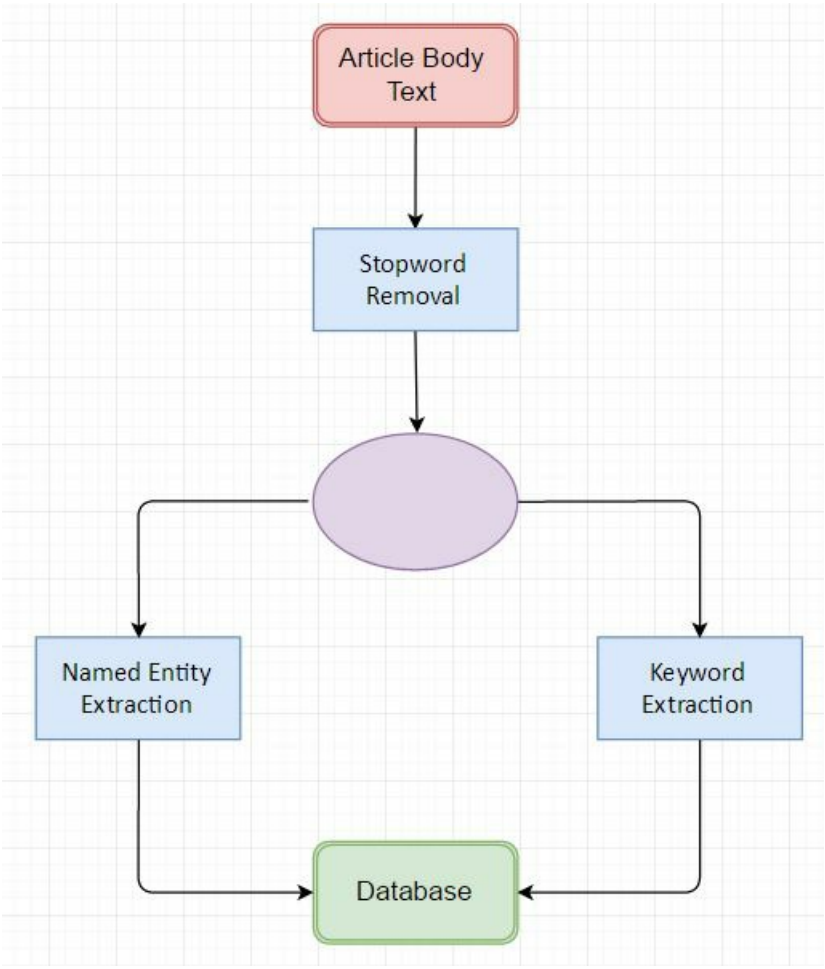
- NLP Workflows
  - Main NLP pipeline from raw body text through TF-IDF.

    Lemmatization and Stemming do similar things, but with vastly different tradeoffs in accuracy vs performance. The choice to use one or the other will only become clear as we explore our data. Any process that has a high computational cost should be stored in the database so it never has to be re-computed (in the end we may choose to store every stage of the analysis in our database, but the ones shown here for sure should be).



- Named Entity Extraction and Keyword Extraction

  Note that this is shown in a separate diagram for the sake of clarity but these processing stages will probably be incorporated into the larger pipeline shown above for sake of simplicity in data pipelining.



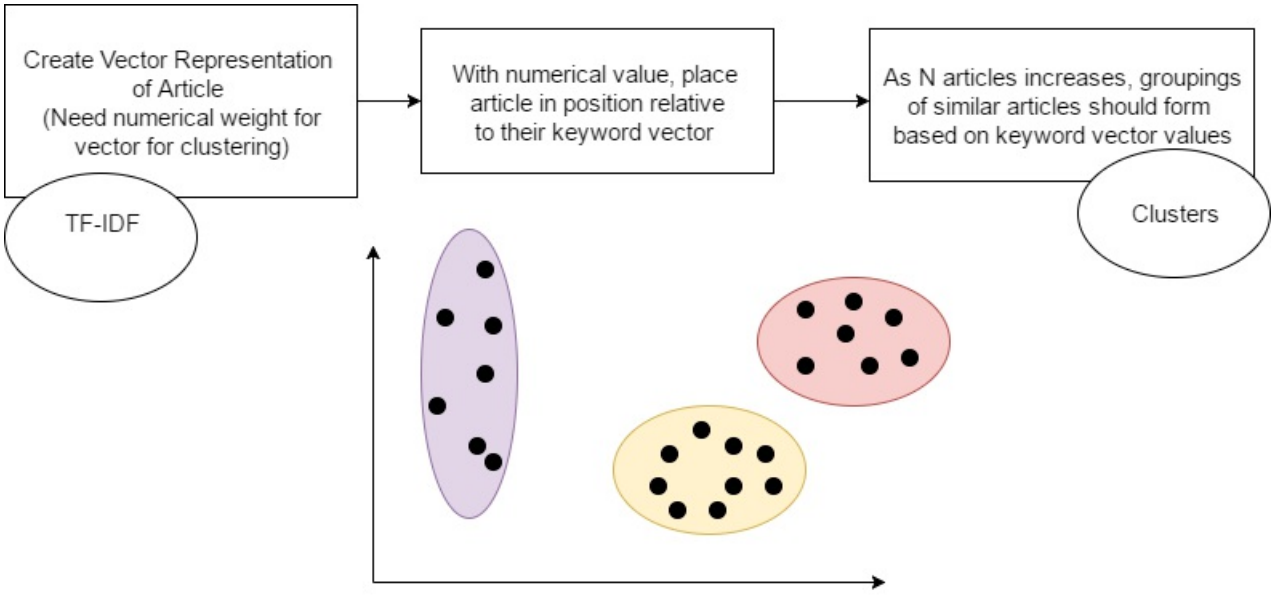
Machine Learning Analysis

This section will briefly outline the machine learning techniques we intend to use within this project.

- **Unsupervised Learning**
  - **Clustering**

    At a high level, clustering is simply a process of letting data group together by placing points near those other data points that are more similar; of course more different data points would be farther away from each other. This allows the data to create clusters, simply meaning groups of similar data points. In our case, we will use K-means clustering, a technique more thoroughly discussed in the research paper about this topic. The most important aspect is that the data clusters are based purely on the similarity of their keywords, and at a later step the clusters will be characterized using training data.

    Another important thing to mention is that articles can overlap with their topics, so it is important not to try to over distinguish clusters from each other, as overlap is expected with this type of dataset. K-means clustering might need some modifications in order to ensure the overlap is not lost (if it is significant).



- **Supervised Learning**
  - **K-nearest Neighbors and Distinguishing Clusters**

    We will be employing the classic KNN technique within our project in a unique way. Once we have created clusters from the data, we will supply training data, and based upon that training data and its nearest neighbors, we can hopefully somewhat classify our clusters based on the training data and its placements within. KNN will be the way we calculate the proportion of our training points within clusters. For further details about KNN check the research paper related to clustering.

Further, once we have chosen a fixed set of so-called 'topic clusters', we can grow these clusters as we add more datasets, hopefully creating more diverse and accurate classification of new articles as the model improves through laws of big numbers in statistics.

- **Sentiment Analysis**
  Sentiment analysis is a supervised learning process of classifying text as having either a neutral, positive, or negative sentiment. The classic example analyzes movie reviews. A sentiment analysis classifier is trained on a subset of data from a dataset containing reviews of movies that have already been rated as being good movies or bad movies by viewers with the goal of being able to classify new, uncategorized reviews as being positive or negative. The classifier is then tested on the remaining subset of the data to check for correctness of predictions.

  The accuracy of sentiment analysis is heavily dependent upon the training data provided to it, especially in its similarity to the target data. Research shows that even human raters agree on text sentiment only 79% of the time, so an algorithm that is 70% accurate is doing a very good job.

  There are two main challenges in analyzing sentiment from news articles:

  1. Reporting in general purposefully employs a neutral style of language and it may be hard to extract meaningful sentiment classification results when the authors are making a conscious effort to employ neutral language.
  2. Finding a labeled dataset similar enough to the project's data so a sentiment classifier trained on the labeled dataset can provide meaningful results when analyzing the project's news article dataset. There are a large number of sentiment datasets regarding things such as Tweets, movie reviews, and product reviews, but datasets labeling news articles might be difficult or impossible to find so the developers might have to creatively combine more generalized sentiment analysis techniques or label some of the data by hand.

  To address the potential issues raised in point 2 above, NLTK offers some generalized text corpora labeled with sentiment that could be used as training data. These include the opinion_lexicon containing a list of positive and negative words in English, and the sentence_polarity corpus containing over 10,000 sentences tagged as positive or negative.

  Finally, Python's NLTK and Sklearn libraries offer machine learning sentiment analysis algorithms. NLTK's NaiveBayesClassifier is the most widely used.

  Pseudocode for sentiment analysis:

```
from nltk.classify import NaiveBayesClassifier

# import a dataset labeled by sentiment
# Separate the dataset into training data (call it training_set) and testing data  (test_set)
# Majority split should be training data, i.e. 75% training/25% test

classifier =  NaiveBayesClassifier.train(training_set)

# Compare results with test_set using comparison function
nltk.classify.util.accuracy(classifier, testfeats)
```

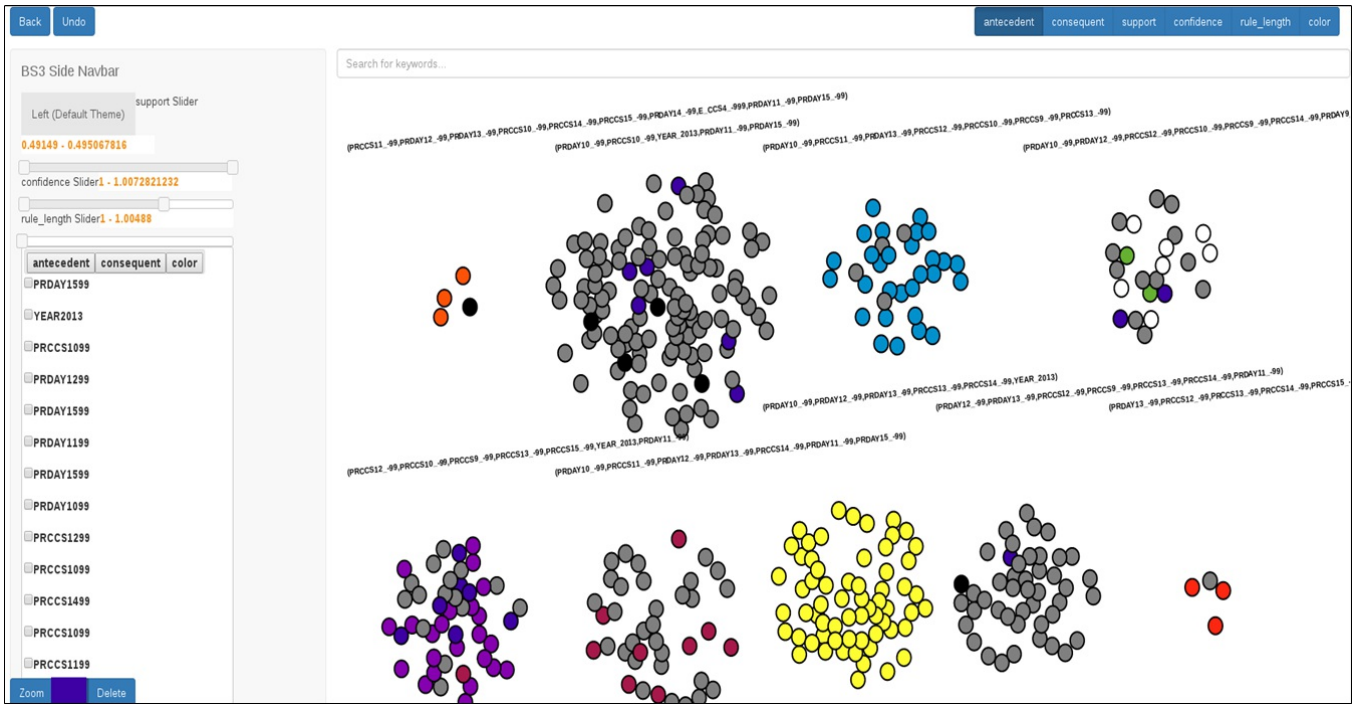
Formal Procedure for Taking Text to Valid Topic Clusters

- (1) From the entire dataset, word frequencies will be calculated. Stop words will be removed from this list of words, either through pre-processing of the text or by removing them from the list of word frequencies.
- (2) Keywords will be selected by choosing a subset k of the most frequent words; these selected keywords will be used for TF-IDF analysis on each article. This will represent the article as a vector of keywords.
- (3) Using cosine similarity or Euclidean distance, similarity will be measure between articles. Those more similar will be grouped closer, less similar further.
- (4) Once the entire set of articles is measured and positioned relative to other articles, areas of highest density should naturally occur, indicating similar articles in that area. Using KNN-like measurement, and training datasets pre-determined for some topic, the clusters will be classified and identified as representative of those pre-determined topics.
- (5) Using this model, repeated training can be supplied to improve it, and as users submit articles, their submitted articles can be classified and described, then utilized back into the model to improve it as well.
- (6) Once a similarity classification system is in place, we can attempt to extract meaningful sentiment classification at many different levels including by article, by topic cluster, by news outlet, etc.

Phase IV: Web Application

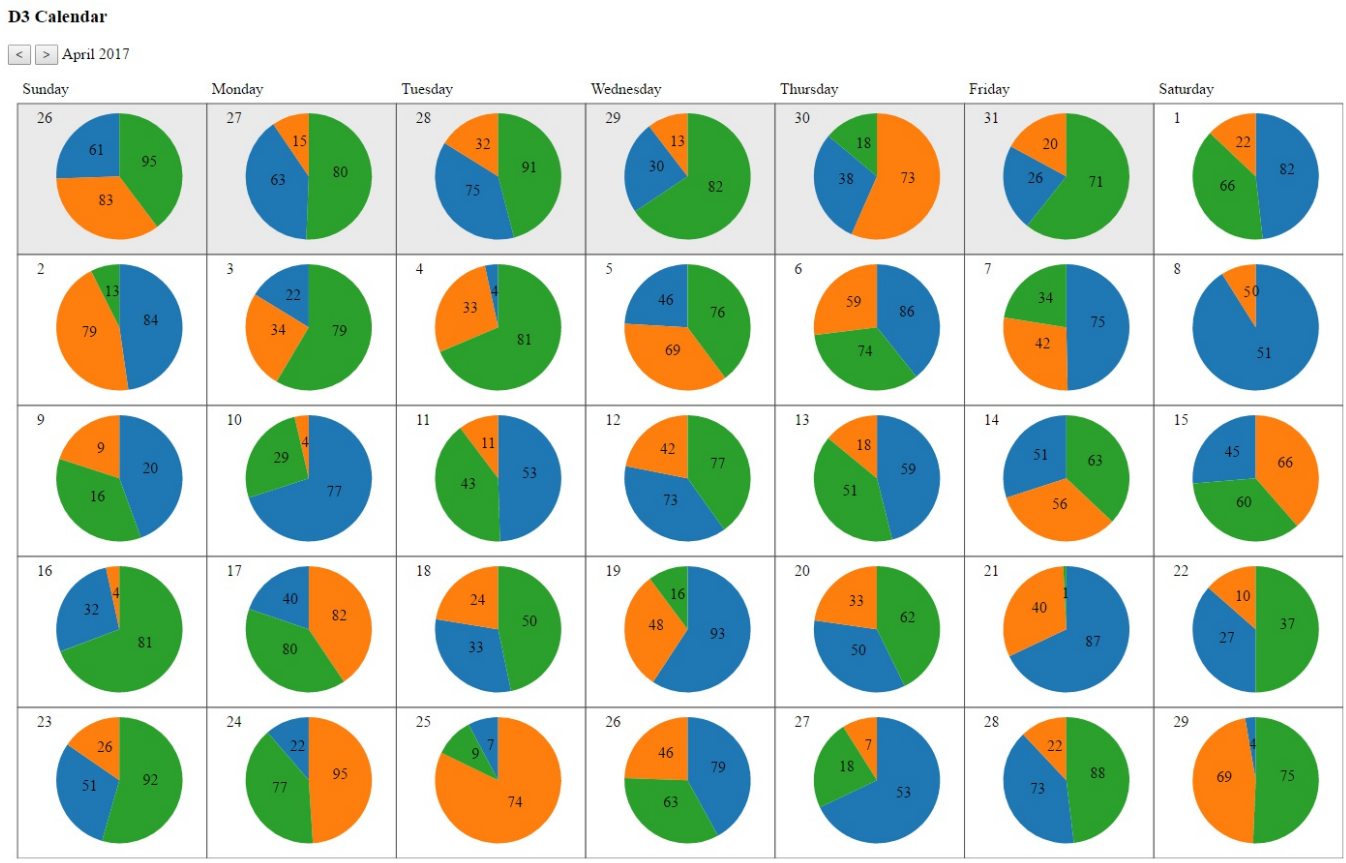
- Web framework
  - **Flask**
    - Allows communication between our Python-based back-end and our front-end web application.
    - Flask is a micro framework which means that it has a simple yet extensible core. This is important for a computationally intensive project because any overhead incurred by functionality not being used would be detrimental.
    - One of the easier to learn and use frameworks with a low amount of organizational overhead compared to many other frameworks, allowing for rapid integration and expansion.
- Languages
  - **HTML**
  - **JavaScript**
    - D3.js library will be used for visualizations of data analysis provided by processing done in Python on the back-end and communicated via Flask.
- Data Visualizations
  - **Clustering Visualization/Explanation**

    In order to provide meaningful information to end users, and for purposes of measurement and accuracy and interacting with the clusters of data, creating a visualization tool for this clustering could serve both of these purposes. This graphic could show density of clusters, count of articles within, boundaries of clusters, and/or any information that is determined to be useful in relation to the grouping of the articles by similarities.

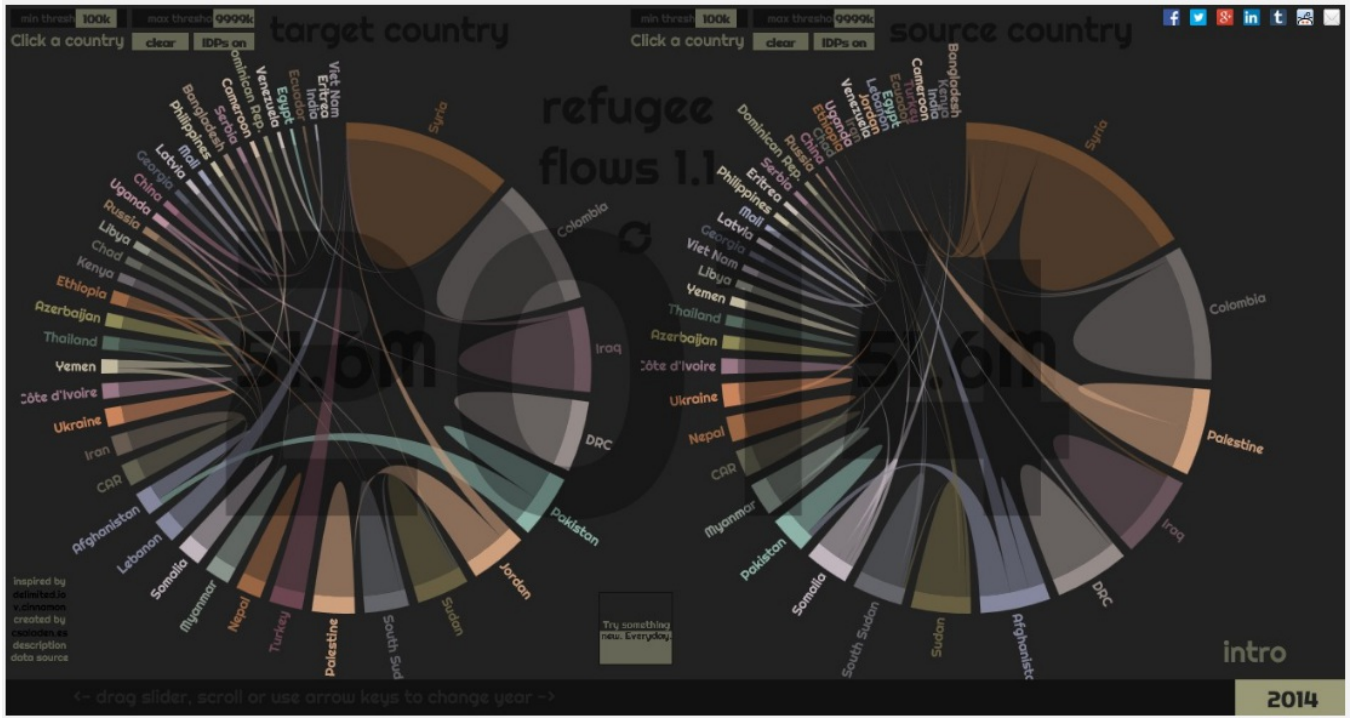


In this example, node size represents confidence, cluster represents rule part groupings, color represents something else and transparency could be another stat if it's not overwhelming.

- **Examples of Statistics-based Visualizations/Explanations**



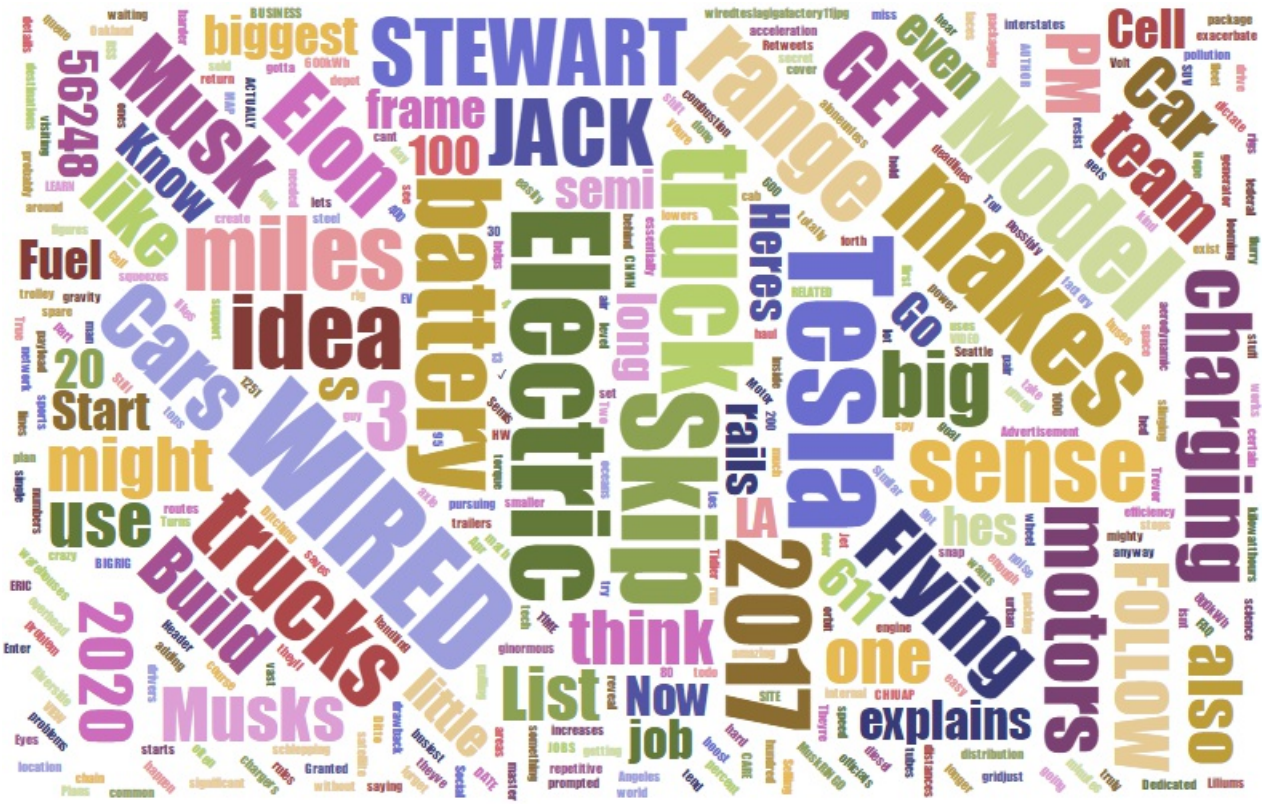
By day we can show the amounts of the pie graph based on topics, or based on sentiment towards certain topics.



This could use news sites around the and the relations could be be cross site topics.

More information

- **Wordcloud example/Explanation**

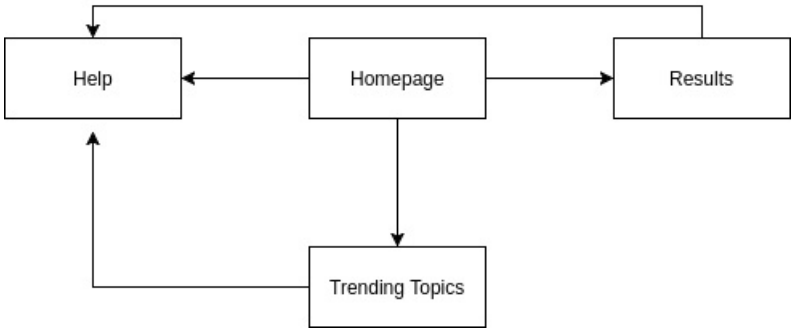


Size will be representative of the word count, color will indicate sentiment and orientation will represent something else.

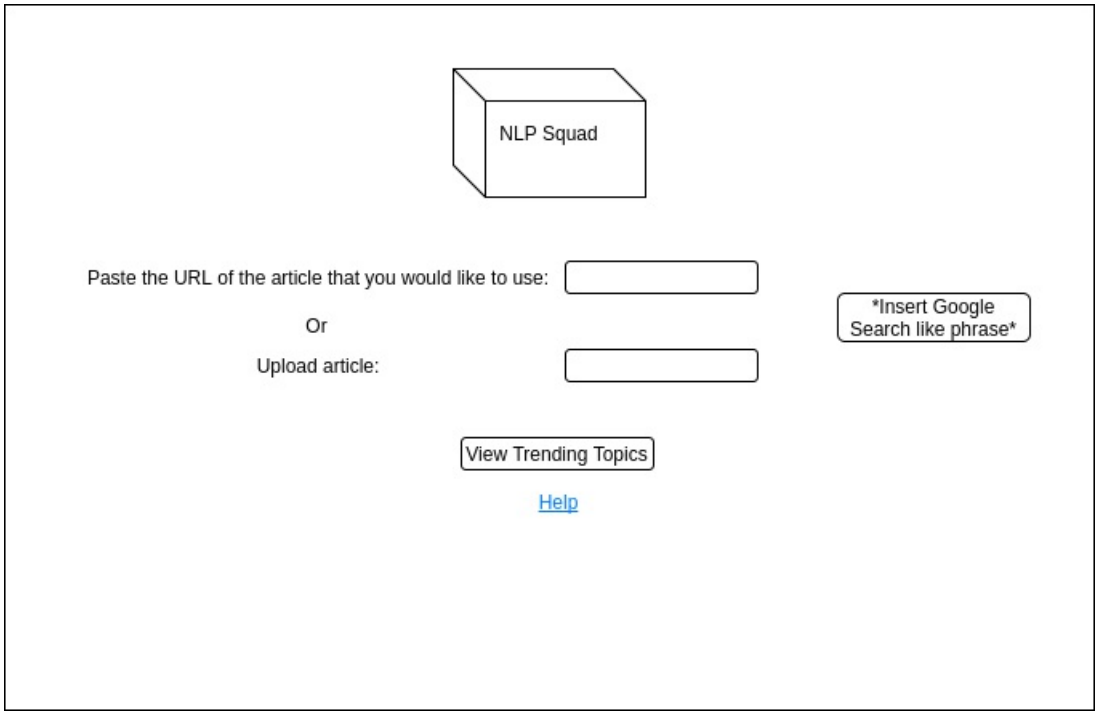
Link to Potential Python Library

- User Interface Diagrams

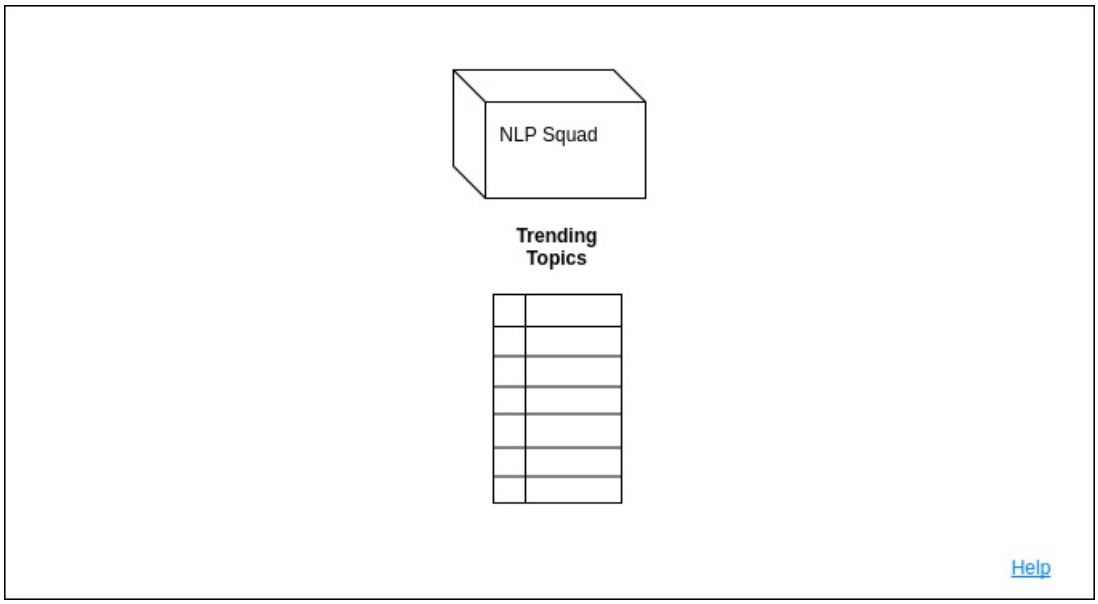
Site Flow



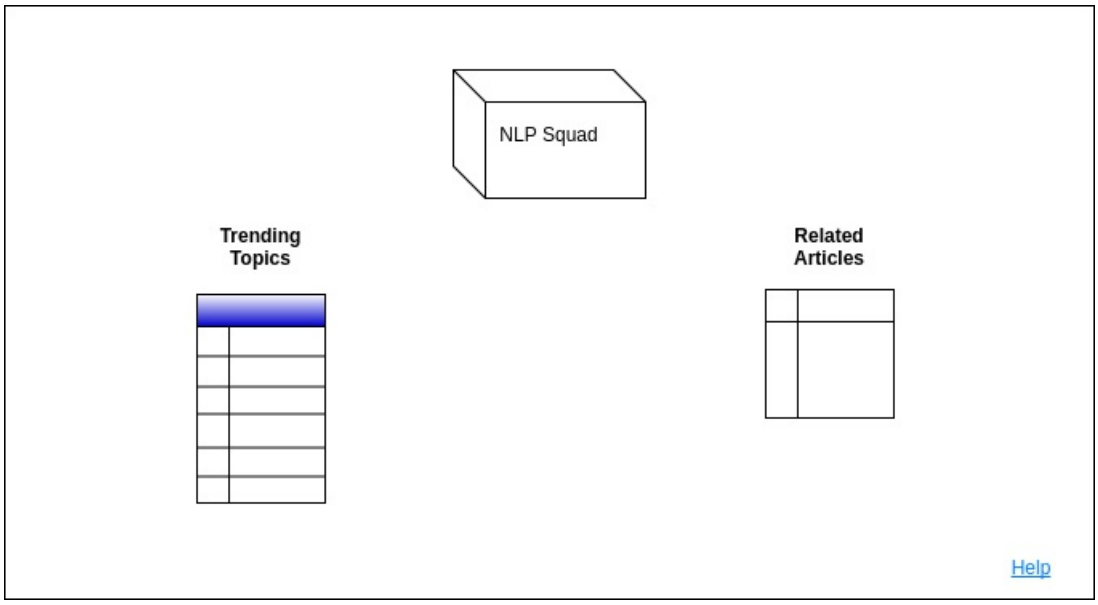
Homepage



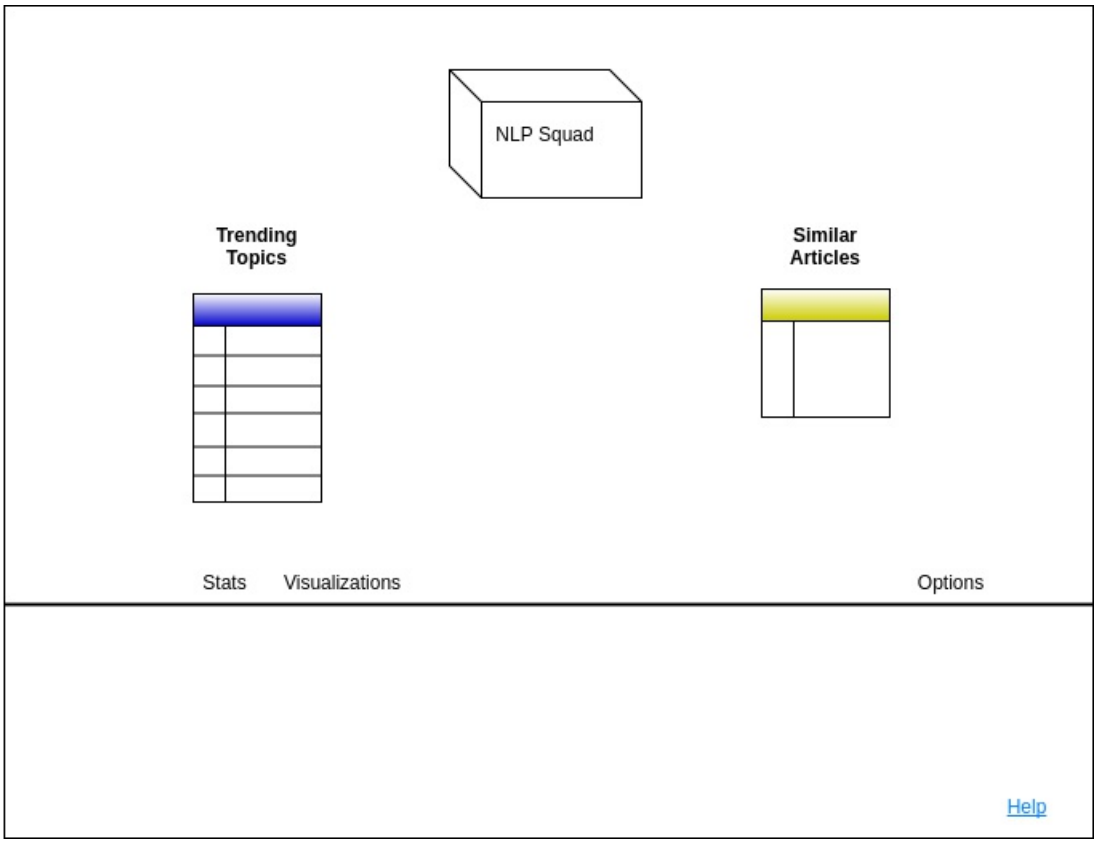
Trending Topics



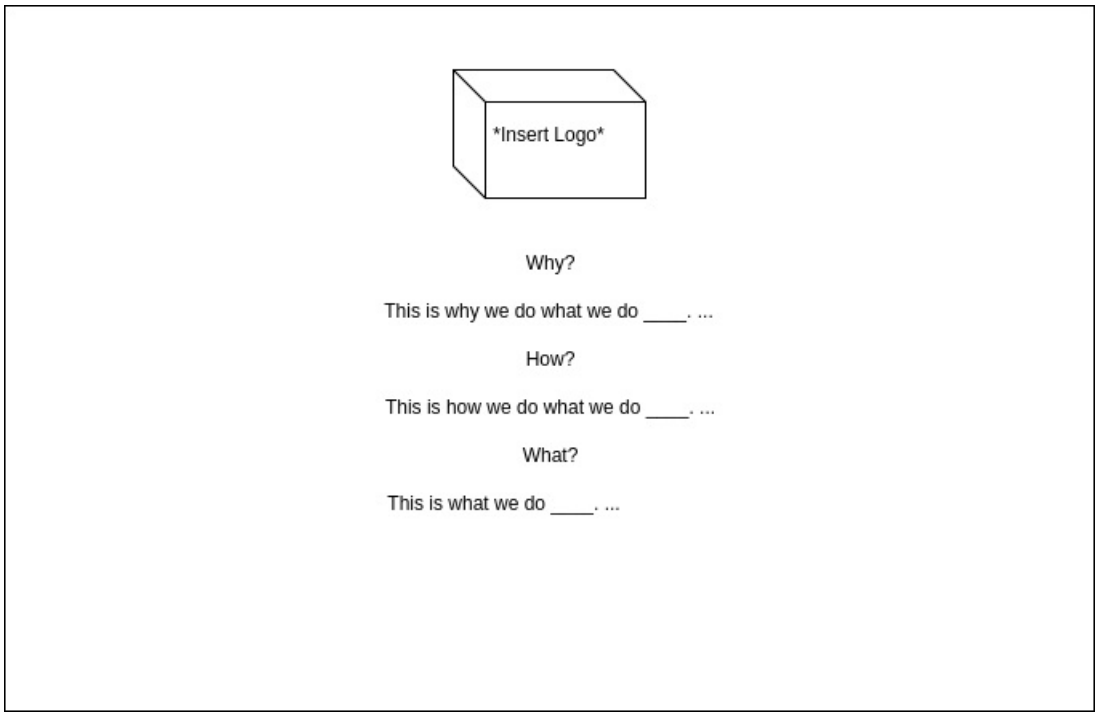
Topic Chosen



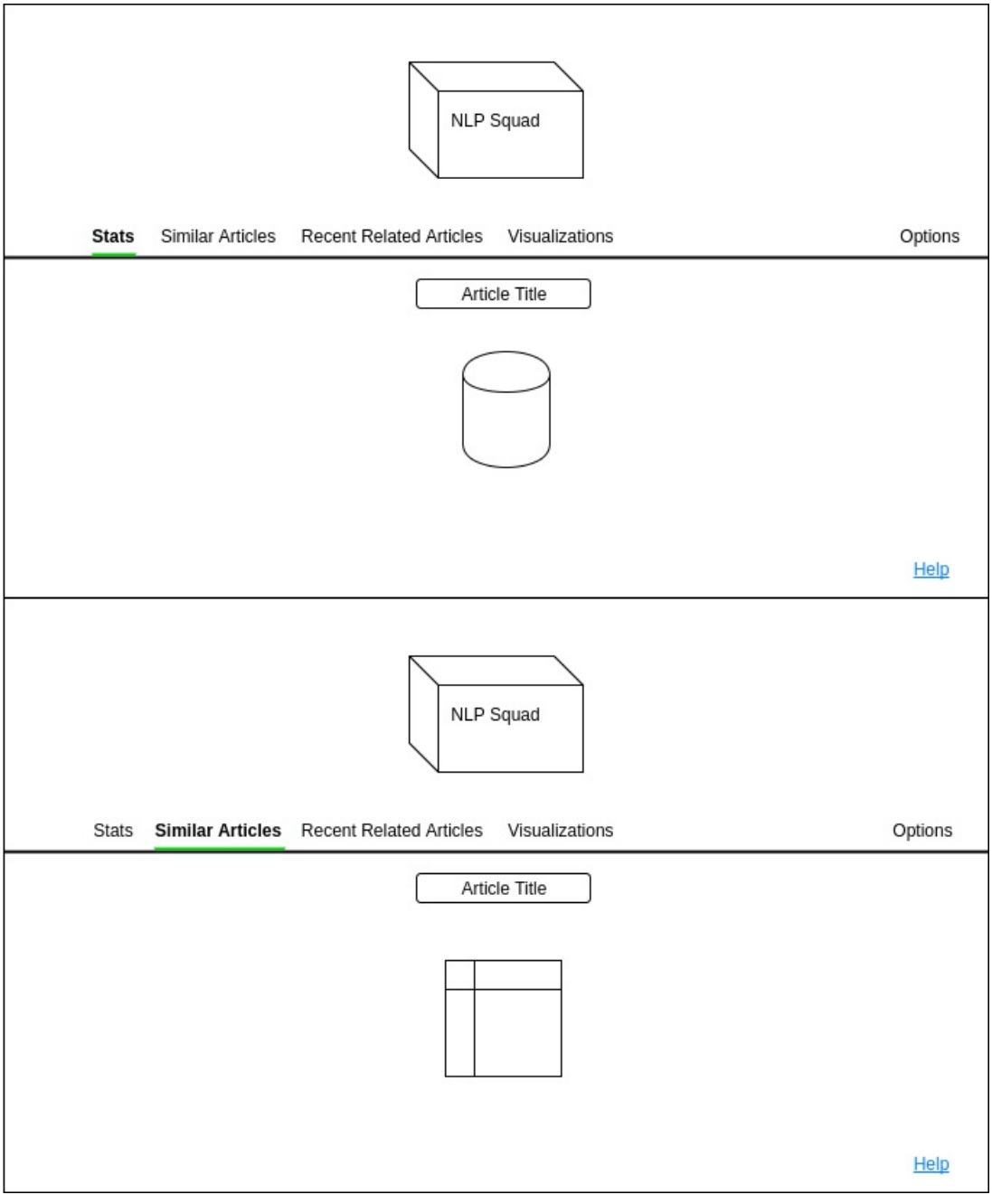
Topic + Article Chosen

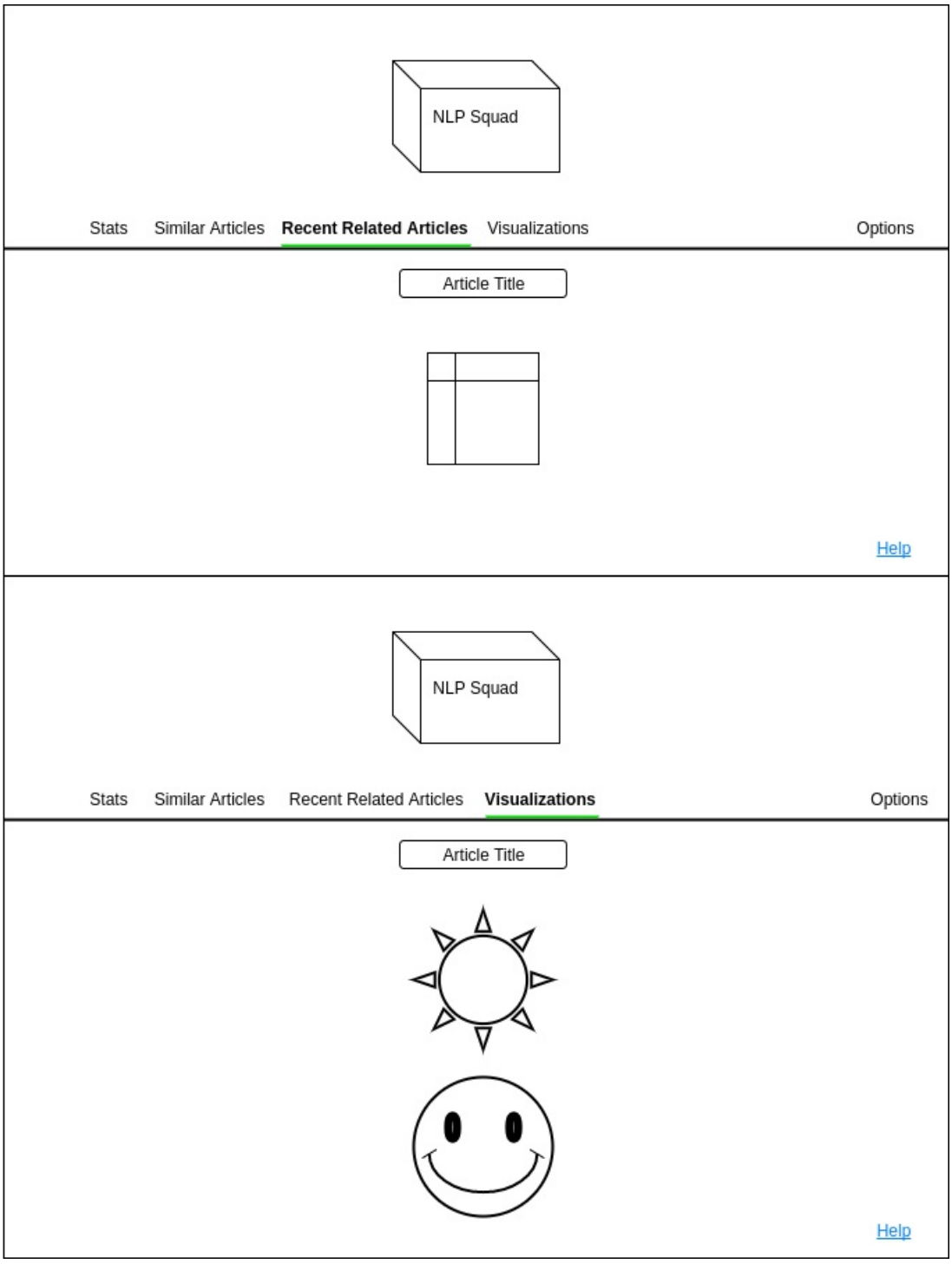


Help

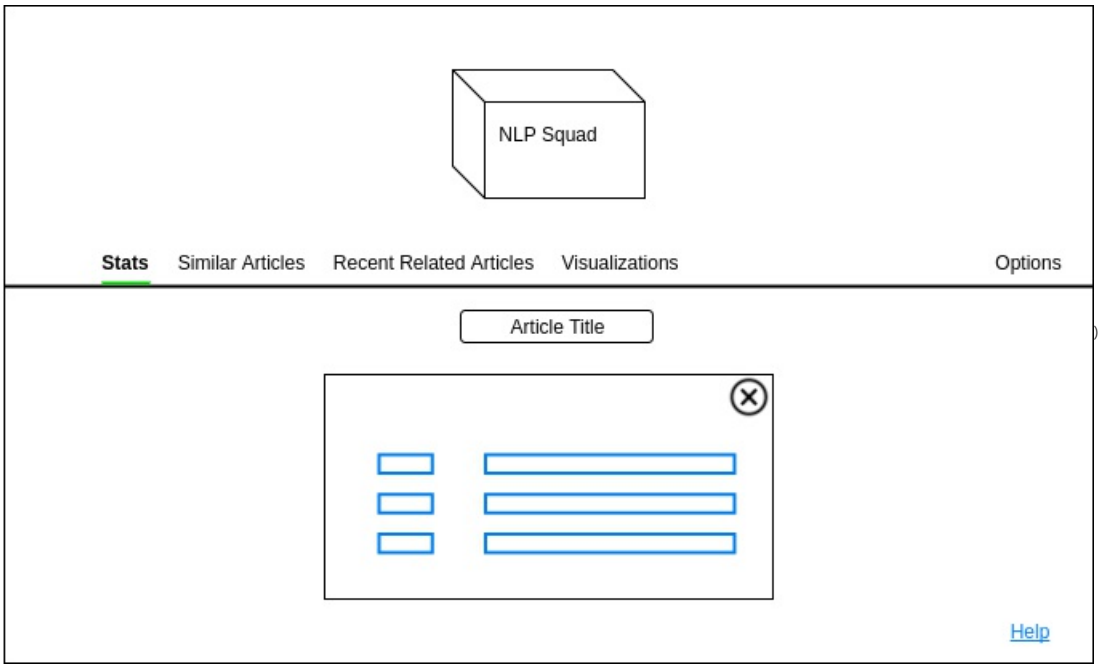

Insert Logo
Why?
This is why we do what we do ____. ...
How?
This is how we do what we do ____. ...
What?
This is what we do ____. ...

Results


NLP Squad
Stats
Similar Articles
Recent Related Articles
Visualizations
Options
Article Title
Help
NLP Squad
Stats
Similar Articles
Recent Related Articles
Visualizations
Options
Article Title
Help


NLP Squad
Stats
Similar Articles
Recent Related Articles
Visualizations
Options
Article Title
Help
NLP Squad
Stats
Similar Articles
Recent Related Articles
Visualizations
Options
Article Title
Help

Options


NLP Squad
Stats
Similar Articles
Recent Related Articles
Visualizations
Options
Article Title
Help

Research

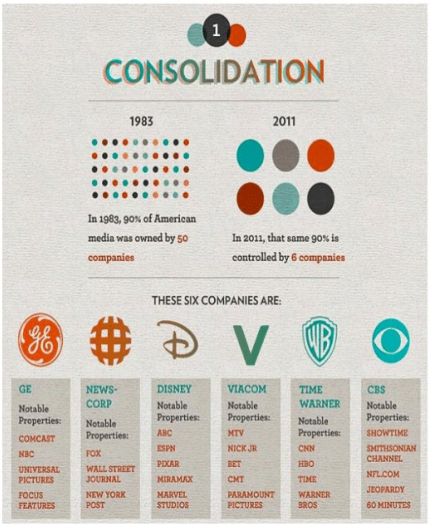
Ali Raza

Gary McKenzie

Capstone I

20 April 2017

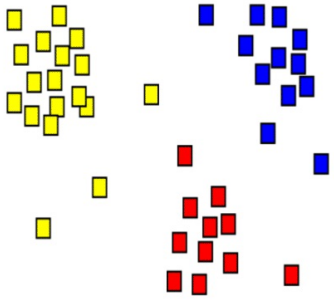
Classification and Analysis of News Articles using Machine Learning



The Illusion of Choice is a phrase that refers to the monopolization of major services in a capitalist society. This phenomenon can be observed in the production of media in the United States. In 2012, Business Insider reported that 90% of media in the US is controlled by 5 companies [2]. Following Kellyanne Conway's use of the phrase "alternative facts" to defend misinformation disseminated by the White House, there exists a legitimate concern regarding the integrity of available news. Irrespective of political ideologies, it is evident that there is a pertinent need to analyze news content in order to make meaningful inferences and discover hidden trends and patterns. There have been many impediments to analyzing news in the past. Primarily, we have to analyze a massive amount of data before we can make any meaningful inferences. Furthermore, human analysis of news will always be conducted with bias, whether intentional or not. Advancements in distributed computation have made it possible for Computer Scientist to analyze massive data sets. Furthermore, various machine learning algorithms have made it possible for us to study media content while limiting bias. In this paper, we will

focus on techniques for classifying news articles, an essential pre-processing step that will allow us to discover similar articles. We will then discuss whether it is possible to perform further analysis on articles that fall within a particular classification to discover meaningful trends and patterns pertinent to a specific classification.

Designing efficient algorithms that group objects into a set is known as clustering, a focal point of machine learning research. The primary goal is to create sets such that objects within a set are similar to each other, based upon some metric. In exploratory data mining, clustering is used as a pre-processing tool. To understand its necessity, consider the image above. Suppose each square represents a news article. The clusters (distinguished by the color of the square) group articles that are similar based on a criterion. For example, if the goal of our analysis is to identify how prevalent buzzwords are in articles based on their publishing source, the criterion would be the publishing source. Why would this step be necessary? Since each article (regardless of publishing source) will have a different percentage of buzz words, we would need to aggregate the percentage of buzzwords for all articles. However, it does not make sense to collectively analyze articles from all sources because our goal is to find how prevalent buzz words are for each source. Here, we can use a clustering algorithm to separate news articles by source, and then subsequently aggregate and measure the average percentage of buzzwords for all articles within a cluster. Clustering is essential in data analysis because data in real life forms “natural categories”. For example, news

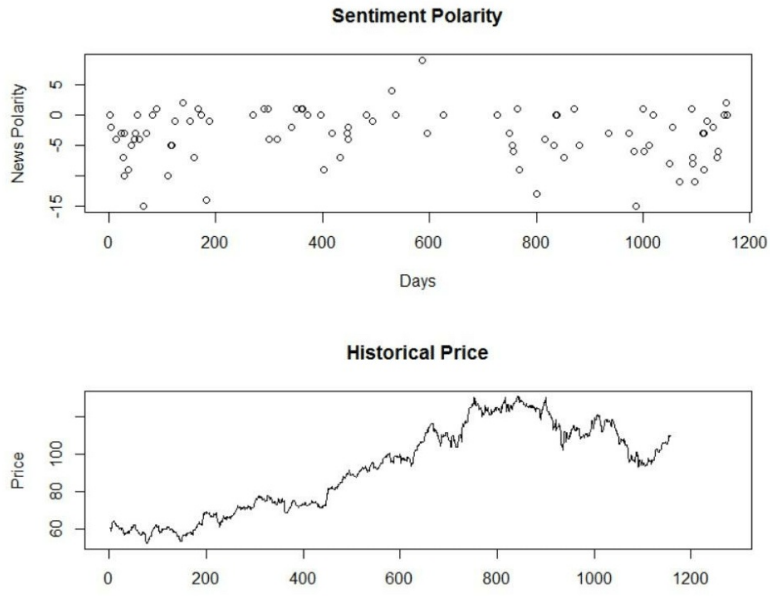


articles can be clustered by author, source, topic, emotion conveyed, tone, etc. However, when data is collected, a computer cannot distinguish whether two articles are the same topic. This is where clustering algorithms come into play: they identify and form the aforementioned “natural groups” so our analysis can be focused upon the topic we are interested in.

Clustering is a form of unsupervised machine learning because clustering algorithms work without a predefined knowledge of the dataset. In contrast, there is another methodology: supervised learning, which generates inferences from trained data sets. In 2009, MIT students Dennis Ramdass and Shreyes Seshasai attempted to classify news articles using Naïve Bayes Classification. A Bayesian Classifier is an approach that makes assumptions about the generation of data, and then subsequently conceives a probabilistic model that embodies these assumptions. Bayesian classifiers use supervised learning on training sets to approximate the parameters of the generated model. Classification is performed with Bayes’ rule by selecting the category that is most likely to have generated the example. The naïve Bayes is the simplest classifier: it assumes that all features are independent of each other within the category. This is consider naïve because it is false in real-world applications. Despite this assumption, the naïve Bayes performs well in classification and is simpler to implement because the independence assumption allows for features to be learned separately. Ramdass and Sesahsai found that “with the right feature selection and a large enough training size, [it is possible to] create a classifier to classify documents with an accuracy of 77% into their respective sections” [3]. Ramdass and Seshasai’s work is one of many examples of using machine learning to categorize news articles. Their

concluding results support that it is possible to classify text documents into categories at an acceptable level of accuracy.

Having discussed the essential pre-processing step of clustering data in order to categorize articles and having provided evidence that machine learning is capable of generating categories at a respectable accuracy, we transition to the analysis of articles once categories have been formed. In 2016 researcher from KJSCE, a university in Mumbai, performed sentiment analysis on news articles to predict stock trends. The methodology was as follows: The documents were tokenized into word vectors which were subsequently compared to pre-built dictionaries of positive and negative words. The difference between the frequency of positive words and the frequency of negative words was used as a polarity measure for the news articles. They concluded that the sentiment of an article was able to accurately predict stock trends with up to 92% accuracy [1].



In conclusion, machine learning has been a promising tool in analyzing news articles. Both supervised and unsupervised machine learning algorithms can be used to categorize news articles. Furthermore, once categorization has been achieved we can use various techniques, such as sentiment analysis, to predict trends and patterns based upon data gathered from news articles.

Data Analysis with Machine Learning and Natural Language Processing for Web Presentation: Language Choice

By: Justin Renneke

Capstone I

4/13/17

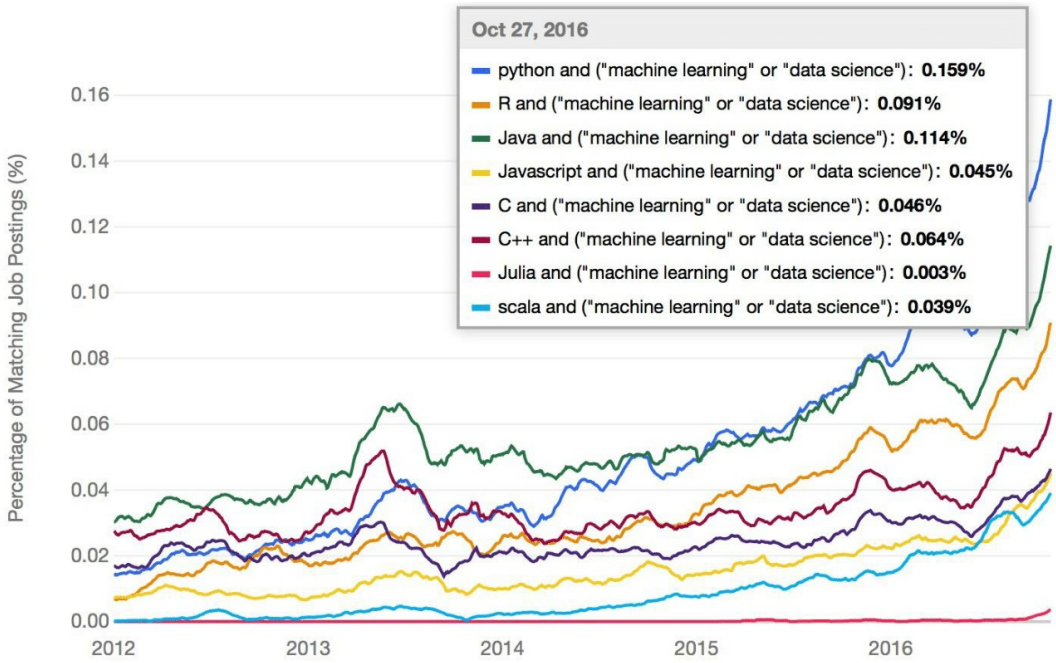
Choosing the right programming language for a project is like choosing the right construction material to construct a building. Wood and steel are both good building materials in their own way and both have been used to make impressive, well-engineered structures, but each has its own unique strengths and weaknesses. Choosing to build a skyscraper from wood instead of steel has the obvious downside of putting a relatively low limit on how tall it can be built. Building a family home from steel instead of wood has downsides such as higher initial cost and the requirement of more highly skilled and specialized contractors to build it, as well as more subtle issues that would only appear later in the home's life such as greatly reduced insulation capabilities due to the temperature conductivity of steel. However, by and large, a home is a home and a skyscraper is a skyscraper to the occupants inside. Similarly, the typical end-users of a well-designed software product will never know or care what language was chosen to construct it, but this choice has major implications for those who will code it and must live with its foibles and limitations. Just as in the choice of construction material, choosing a language well-suited to a project will reap many benefits for the developers of the project such as: facilitating the full realization of the project's scope by removing or avoiding limitations inherent in other languages; reducing time investment by accomplishing the same goal with a technology that is easier to use; or decreasing the cost of future maintenance. The problem addressed in this paper will be to choose the most suitable programming language for the back-end portion of a project involving the analysis of textual data for presentation in a web application.

The first of these two challenges - performing back-end data analysis on a large amount of text - has several key characteristics to consider that are specific to the nature of the

challenge. The first and most important factor to the success of this task is: the data analysis will involve statistical analysis and machine learning, but the developers do not have the time or expertise to create machine learning algorithms from scratch, so the language should have pre-existing machine learning libraries available. Second, the data will need to go through layers of cleaning and feature extraction using natural language processing, so support for this must exist. Finally, a huge amount of textual data will need to be processed, so performance is an important consideration. The second challenge - providing visualizations of the data analysis within a web application - will require dynamic interaction between the back-end and front-end systems in the project to allow for graphing and charting the results within an interactive web interface based on HTML, CSS, and JavaScript. In addition to these task-specific considerations, there are also some more general concerns to keep in mind. The first of these is that the developers are college students who have a relatively limited repertoire of language experience. Learning a difficult language such as C++ from scratch is not feasible for this project. In addition, the scope of this project is very ambitious considering the skillset of the developers, yet it must be completed within a timeframe of a few months.

Given the heavy focus on data analysis and machine learning, the initial language candidates can be narrowed down to three for in-depth analysis: R, Java, and Python. This decision is supported by an article on the IBM DeveloperWorks blog[4] which explored the question of which languages are the most popular to use for machine learning and data science

by searching indeed.com job postings and the results can be seen in the chart below.



An analysis of each of the factors integral to the success of the project as outlined above done within the context of these three candidate languages will reveal the best tool for the job.

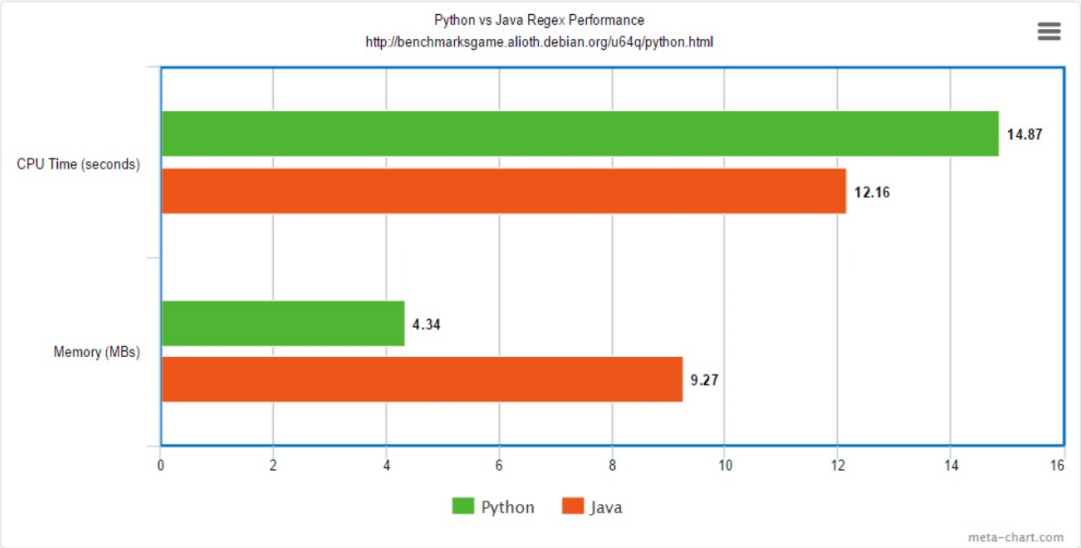
The first required capability - the availability of robust support for machine learning via libraries - is well supported by all three candidates as expected given that they were chosen based on this, the most important criteria for the success of the project. According to a highly upvoted post [5] on Stackoverflow that compares the machine learning libraries of R and Python, R has over 5000 (somewhat fragmented and overlapping) libraries that support machine learning and data analysis while Python has several comprehensive libraries such as Pandas, Scikit Learn, Scipy, and Matplotlib that offer similar capabilities. Java too, provides very good machine learning libraries ranging from Weka, which provides not only a machine learning

API but also a graphical user interface to facilitate experimentation, to Java-ML. This capability is essentially a tie across the three languages as each language provides library support for the full range of the data manipulation and machine learning analysis pipeline from pre-processing and feature generation to analysis using clustering, classification, and other techniques.

Secondly, the textual nature of the data being processed and analyzed requires that the language provide extensive support for natural language processing (NLP). This highly regarded reply [6] to a Stackoverflow question asking if Python or Java was a better choice for an NLP project states that both of these languages offer excellent support for this task. Python has a large and full-featured NLP library in the form of the Natural Language Toolkit (NLTK) library and Java offers multiple powerful NLP libraries such as the comprehensive OpenNLP library and the highly regarded and flexible StanfordNLP library. R, on the other hand, appears to lose out in this category. While R does offer support for NLP in the form of its Text Mining Package (tm), this blog post[7] from a data science course teacher suggests that “R's primary NLP framework (the tm package) [is] significantly more limited and harder to use [than Python’s NLTK].” Anecdotal evidence gathered from Google searches seems to support R’s inferiority in NLP. While there are a great many results for Python and Java in regards to NLP, there are relatively few for R NLP and those that do exist are lesser quality and scope, at the very least indicating that the support community for R NLP is not up to par.

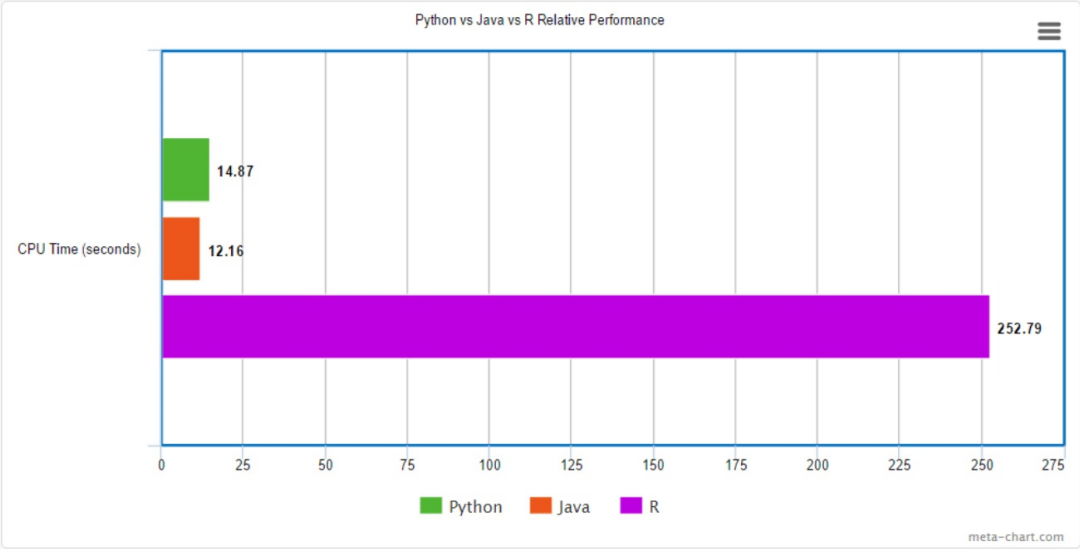
The third consideration, raw performance of the language when processing strings, is important due to the sheer amount of data that will need to be processed. This can be difficult to assess to a certain degree because it is highly dependent on the quality of effort spent to write optimized project code. It also depends on libraries leveraged - some languages have

libraries that interface with C code to do heavy computations, thus presenting the opportunity to achieve a significant performance advantage over the base language - and the type of computation, among other things. There is also the issue of choosing which metric to emphasize: processing speed, memory, or something else. The following chart from a website [8] comparing performance metrics across several languages shows the results from Python and Java when evaluating a Regex expression (lower numbers are better).



Unfortunately, similar data was not available for R, but research into R's performance relative to other languages shows a general consensus among data scientists that R suffers in comparison. A companion website [9] to the R textbook *Advanced R* states that "R is not a fast language. This is not an accident. R was purposely designed to make data analysis and statistics easier for you to do. It was not designed to make life easier for your computer." Another article [10] from a blog dedicated to discussion of the R language compared R's performance with Python's when calculating prime numbers and found that R was 17 times slower than Python in

this case. Extrapolating this statement to the CPU time metric in the graph above with the understanding that this is only for purposes of visualization due to the significant difference in context, we see the big difference in the chart below.



Based on these results, Python and Java have similar relative performance, with Java slightly edging out Python in CPU time, but losing out on memory consumed. R would appear to be a distant third.

Next, since the end results of all of this data analysis is meant to be displayed in a web application, the languages' ability to interface with a web application must be considered. Each of the three candidate languages has support for this to some degree. R has a handful of web frameworks including the Shiny framework that supports the development of interactive web pages and is particularly strong where visualization of data is concerned. Python has many web frameworks from the extensive Django to the lighter weight Flask, and many in between. Java, being the oldest and most heavily used enterprise language, has a wide variety of web

frameworks including Spring MVC, Grails, and innumerous others. R's Shiny provides truly impressive visualizations, but seems to be limited in scope compared to the others as it is mainly used to create dashboard-type setups, providing little interactivity and has a smaller community for support. This project already plans to use Javascript's D3.js library for most front-end data visualizations due to team members' pre-existing experience with the technology, so the main selling point for Shiny is not as important in this case. Python web frameworks such as Flask have a reputation for being relatively easy to set up, for being flexible and facilitating fast development, and for being well-suited to small to medium sized applications developed by small teams, but not as ideal for large teams working together to develop modular, high-performing applications. Java web frameworks such as Spring MVC are generally well-regarded for providing high performance, a truly thorough toolkit given the massive enterprise-level library support, and providing good support for very large teams to work together, but require large investments of time and skill to set up and have high learning curves. These findings are summarized in the table below. R's Shiny, Python's Flask, and Java's Spring MVC are used as general reference points.

General Comparison of Web Frameworks on a Scale of 1 to 5 (5 is best)

	R Frameworks	Python Frameworks	Java Frameworks
Ease of Use	4	3	1
Flexibility	1	4	5
Scope	2	4	5
Speed of Development	4	5	2
Support Community	2	4	5
Total Score	13	20	18

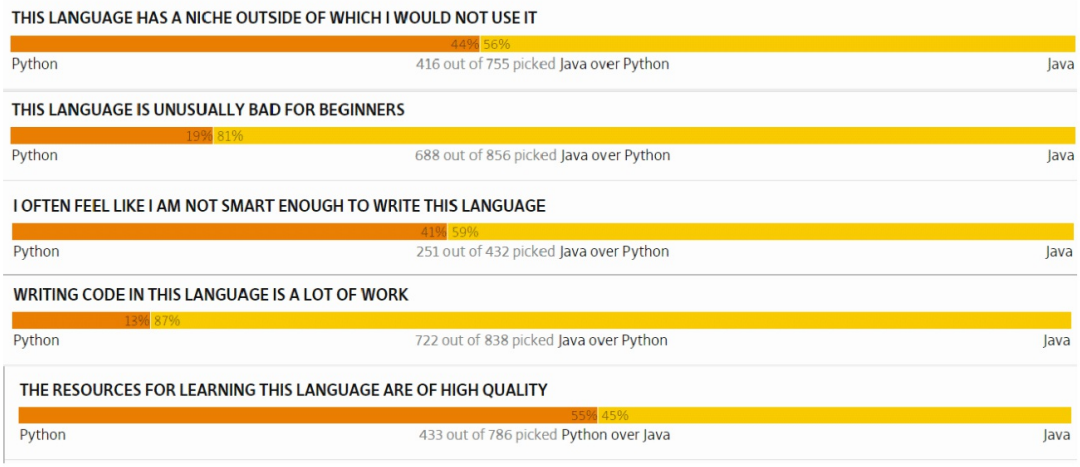
A discussion of Python vs Java web frameworks on Quora [11] supports many of these arguments. In this project's case, Python's web frameworks are the best fit.

Finally, the circumstances under which the project will be developed must be considered. Given that the developers have a limited range of experience with different programming languages, the ease of learning the candidate languages should be considered since some team members will have to learn the language from scratch no matter which is chosen. In addition, this ambitious project must be completed within a few months while the developers are learning as they go, so a robust ability to support rapid development will be essential. R is viewed as a language designed for statisticians more so than for traditional developers. This has led to relatively intuitive syntax but also some counterintuitive handling of traditional programming structures due to this fact. The general consensus is that R is a very specialized language that, while not particularly difficult to get started with, is not as easy to jump into as a more traditionally designed language for people who already know other programming languages and have a pre-existing understanding of program design and logic flow. Also, while R's specialized nature makes it well-suited for machine learning and statistical analysis, that same specialization limits its capabilities in the wider functionality required by the project. Java lies on the opposite end of the spectrum from R. Java has a massive amount of libraries providing developers with a huge range of toolkits. However, it is a highly structured and rigorous language that demands a solid understanding of programming principles from those who would use it. It provides high performance and one of the most extensive object-oriented design paradigms, but inherently has a high learning curve to achieve this. Java is very verbose, has a high overhead to start with, and requires a thorough understanding of object

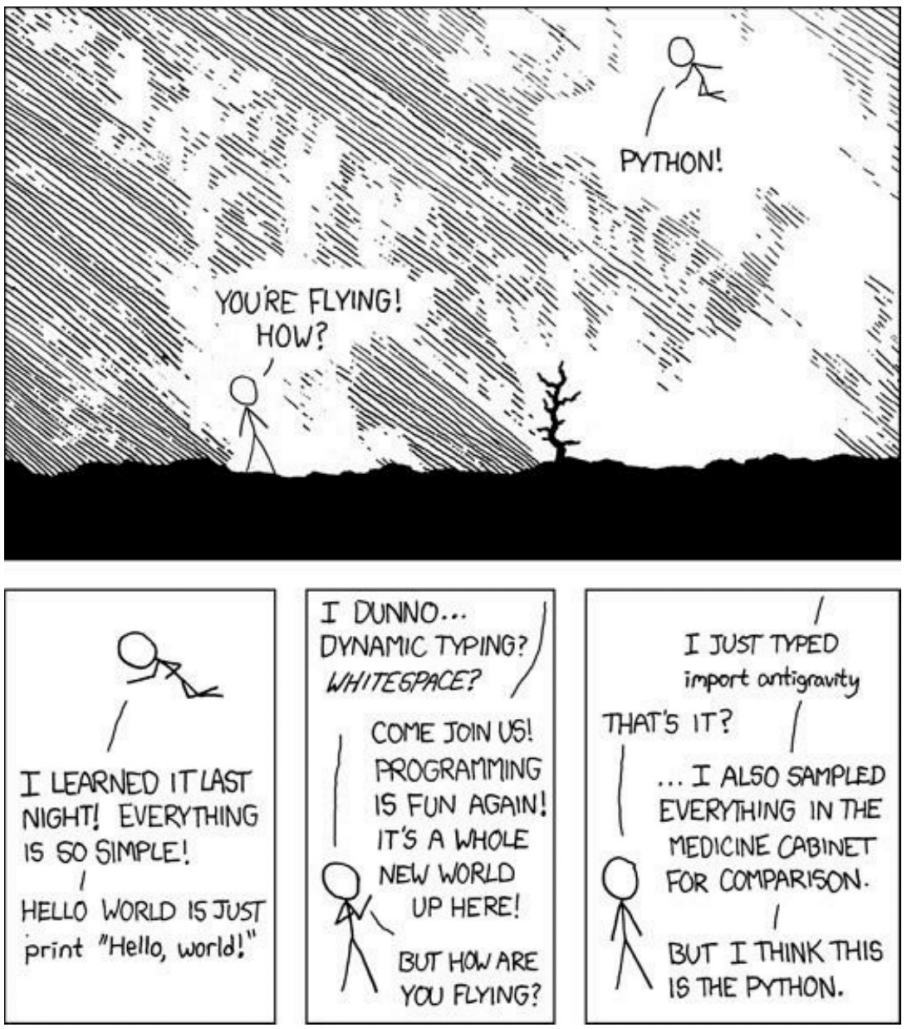
oriented design structures to make use of it. Python, in contrast, is widely regarded as one of the easiest programming languages to learn due to its friendly syntax coupled with traditional programming structures and is famed for allowing developers to accomplish tasks very quickly due to an extensive collection of powerful and easy to use libraries. Users opinions regarding these metrics can be found in the surveys provided here. [12] Some key comparisons can be seen below.

Python vs R:

Python vs Java:



Python's benefits appear to especially shine given the real world constraints of development for this project. The following graphic does a good job of summarizing the communal consensus. [13]



In conclusion, the most suitable language for the back-end data analysis involved in this project is Python. The other two languages considered fall short in some key ways. R is a great language if you are focused on statistical and machine learning-based analysis of numerical data to create visualizations, but appears to be lacking where NLP is concerned. It is also limited

in scope by its lower performance and limited support for web frameworks in addition to being slightly harder with which to work. Java is a very structured, high-performance language with a vast collection of powerful libraries covering everything from machine learning and NLP to web deployment, but these characteristics also cause it to have a high development overhead, giving it a steep learning curve and making it difficult to work with and a struggle to quickly prototype and accomplish goals for the inexperienced. On the other hand, Python provides extensive support for machine learning and natural language processing-based data analysis with libraries such as Scikit Learn and NLTK. Python also provides good performance in comparison to the other candidate languages as well as flexible and relatively easy to use web frameworks for communication with the front-end of the project. Finally, Python is the clear first choice given the time and skill constraints of the development team - it is a relatively easy language to learn and allows developers to quickly accomplish goals with numerous easy to use libraries. R and Java are powerful languages in their own way and the project may even benefit from using some of their specialized libraries which have been discovered over the course of researching this issue. Even then, it turns out that Python wrappers already exist for the most popular of the R and Java libraries, so it seems developers can have their cake and eat it, too, with Python.

Zach Bryant
Capstone 1 – Gary McKenzie
4/21/2017
Research Paper

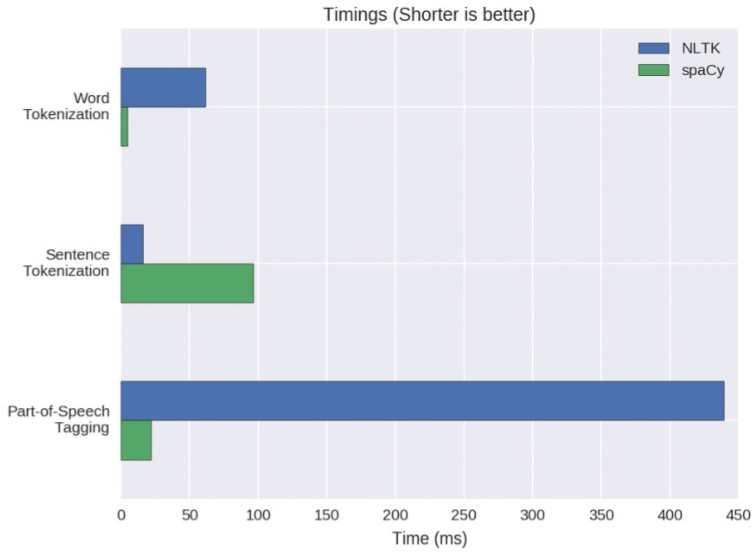
NLP: Natural Language Processing

Ever since computers were invented, computer scientists have been working on ways to model human intelligence. This includes teaching computers how to understand natural language. Natural language is any language that humans use to communicate with each other on a daily basis. Natural language processing or NLP is a field of study in computer science that falls under the category of artificial intelligence and aims to solve the problem of allowing computers to understand, interpret, and speak natural language. This paper will be focusing on the applications of NLP and general information about NLP that is good to know for programming purposes.

So what is natural language processing to begin with? It is the processing of natural language by computers. Natural language is any human made language – not a computer language like C, Java, or Fortran nor a mathematical language like numbers and arithmetic. Any system that takes natural language as an input and does processing on it is a NLP system. For example, when Google Translate is used to translate a word from English to Spanish, this is a natural language process. The input is a human language – English. Some processing is done on it to change it to the desired output – Spanish, a natural language. This is just one of many examples of NLP systems. Some other examples of NLP systems include: automatic summarization, sentiment analysis, speech recognition, and automatic generation of keyword tags. An important thing to note about NLP is that is considers the hierarchical structure of human language – words make a phrase, phrases make a sentence, and sentences convey ideas [14].

Why would anyone want to use natural language processing? NLP has many uses, but more importantly, for a computer to be considered intelligent, it should be able to understand, interpret, and speak natural language. The more that is done to improve the algorithms surrounding natural language processing, the more intelligent computers will become. With this in mind, some algorithms used in NLP include: language modeling, parsing, text classification, and part-of-speech tagging. Additionally, another element that is usually paired with NLP is machine learning. With NLP, it is possible derive more information from the endless data being pumped through the Internet in the form of natural language. Meaning, NLP can be used to look at the tweets on a particular topic and be able to tell if people think positively or negatively about a certain subject. This information could then be used to help improve products and cater more towards customer's needs and wants. NLP could also be used to analyze an article and be able to find other articles that are related to it. Furthermore, NLP can be used to discover patterns in the way that news outlets report information. With fake news being a hot topic today, NLP is becoming more and more relevant to developers who want to fight back against false information.

What tools are developers using to work with natural language processing? One way that a developer can work with NLP is through Python. Python is a high-level programming language that can be used for just about anything. With Python, it is possible to import a library called the Natural Language Toolkit or NLTK [15]. With Python and NLTK, it is possible to program a NLP system to analyze text and produce desirable output. Another Python library that is used for NLP is spaCy, but which library should a developer use? NLTK provides nine different stemming libraries which allows developers to finely tune their models [16]. A stemming algorithm tries to remove suffixes from words in order to find the "root word" or stem of a given word [17]. SpaCy on the other hand implements one single stemmer [16]. This stemmer is maintained and updated by the developers of spaCy. So with NLTK, the developer is left to be creative in developing the best stemmer algorithm by having a choice of nine different libraries. Whereas with spaCy, the developer is given one stemmer algorithm is being constantly improved upon by its creators. This means that a user of spaCy could update to the latest version of spaCy and find that their application has boosted because of it without having to do any extra work. However, there is a downside of using spaCy in that the developer may have to rewrite test cases as spaCy gets updated. Another difference between NLTK and spaCy is that NLTK does all of it processing with strings [16]. Its inputs are strings and its outputs are strings. Contrastly, spaCy implements an object-oriented approach. When spaCy parses text, a document object is returned whose words and sentences are also represented as objects. With each object, there is also a number of attributes and methods that can be attached. Really, NLTK versus spaCy is a matter of strings versus objects. As far as performance between the two libraries, here is a graph showing the timings for word tokenization, sentence tokenization, and part-of-speech tagging for NLTK and spaCy.



[16]

NLTK out performs spaCy only in sentence tokenization, but this is most likely the result of different approaches. NLTK simply silts the text into sentences whereas spaCy creates a whole syntactic tree for each sentence [16]. SpaCy's method may be slower, but it offers a way to reveal much more information about the text that is being analyzed. SpaCy seems like an obvious choice for a common-use application, but there is also one more thing to keep in mind. SpaCy is limited to English only. However, it is possible to convert the desired text to English through NLP in order to use other languages with spaCy, but it will require some creativity from the developer as this process is not built into spaCy.

What is an example of a NLP use case? As mentioned earlier, NLP is typically paired with machine learning algorithms. This is done because one large set of rules will not always return the most accurate information. With machine learning, a NLP algorithm can automatically learn rules and make statistical inferences by reading text [14]. A general rule is that the

more data that is analyzed, the more accurate the model will be. Social media serves as a great use case of NLP. Companies track what people say online and they can do this through Twitter and Facebook for example. Specifically, if a company wanted to track Tweets to see all the mentions of their brand on Twitter, then the algorithm should start by retrieving each tweet that mentions the brand. Then each tweet should be processed through a sentiment analysis algorithm that outputs a sentiment rating.

In conclusion, natural language processing is a very complex topic in computer science, but if mastered, it provides a path to improving the intelligence of computers to levels that compare to human intelligence. Natural language is just any language that humans use to communicate. Any system that processes natural language is a natural language processing system. Some use cases of NLP are: translation, language modeling, parsing, text classification, and part-of-speech tagging. Computer scientists are interested in NLP because with the creation of the Internet, there is endless amounts of data in the form of natural language that can tapped into with a NLP algorithm. The two best NLP libraries for Python are NLTK and spaCy. The difference between the two comes down to strings versus objects. NLP is often paired with machine learning and a relevant use case of NLP is social media monitoring. Companies monitor social media in order to figure out what people are talking about when it comes to their products and brand name. In the end, natural language processing has many uses and the more that it is researched and developed, the smarter and more intelligent computers will become.

Kurt Bognar

Date Due - 4/19/17

Capstone 1

Research Paper

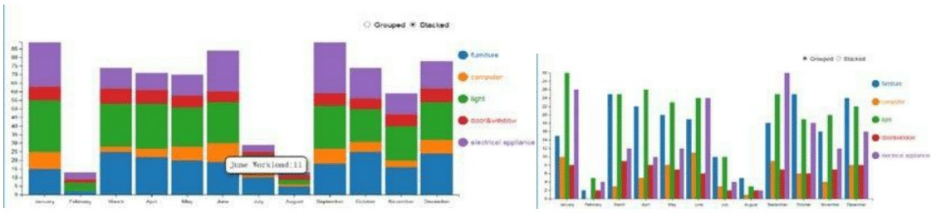
D3 Powered Visualization

Existing methods of visualizing information are sophisticated; however this has yielded many different paid visualization tools. D3 is an open source visualization that allows many data operations and helps to make data interactive and dynamic. Using data to tell a story would require that temporal data be used. Since databases containing scraped article data from various websites have been built for Capstone earlier in the semester, a temporal visualization would enhance the usability of this project and exploit all elements of the data. The problem addressed in this paper is twofold: what is the best solution to use to visualize data and what are good design principles/elements that should be included in that visualization.

D3

Data Driven Documents (D3) is an open source JavaScript library that promotes easy visualization. [20] Even though Tableau makes visualizing data more simply, since it is not free it does not promote the wide adoption of this projects output. Infographics also could be a good way to tell the story attempted by this project, but since they are not typically dynamic they again do not allow for wide spread application and adoption. The ability to dynamically show what an article contains and tie that statistically to a word cloud is ideal. Adding interaction on top of that through D3 allows the user to increase their relationship to the output of this product. For

example, a user may want to see how CNN and FOX have differing relations towards the topic of “alt-right” during the months leading up to the US’s 2016 presidential election while another user wants to see a NBC’s views on “healthcare” for all of 2016. While these users activities are very different, both can be done in D3. D3 achieves this by binding data elements to scalable vector graphics (SVG) before rendering these objects. This allows dimensionality to by dynamically modified before display. Figure 0 shows how a user could view data in two different ways at the click of a box using D3 which prevents additional page loads and wait time while increasing visibility into the data. Having this freedom both in price and customization of views proves why D3 is a good choice for visualizing news articles.

Figure 0: 

Visualization

User interface design has six main principles: structure, simplicity, visibility, feedback, tolerance and reuse. [23] By applying these principles to interactive data visualizations allows for more intuitive use of software. The easiest way to show these principles is through negation and so the following figures are used to show why these principles are necessary. The structure should be intentionally designed and nothing should be random. Figure 1 shows random placement of word, negating any ability to show relationships between words. The interface should be simple and be task oriented. Figure 2 violates this principle by incorporating too many dimensions into each unit it displays. Visibility means that a user should get the things they need on-demand and no sooner. No materials should be redundant and overlapping is discouraged.

Figure 1 violates this but could've avoided this design flaw by rotating objects or using a packing algorithm. Since infographics do not incorporate interaction, they do not incorporate a feedback loop unlike visualizations. The visualization needs to have sufficient error checking so that it is stable and accounts for times when the article extraction tool is down or if a website went down. Since the visualization will dynamically pull data from a database, the reuse principle will also be fulfilled.

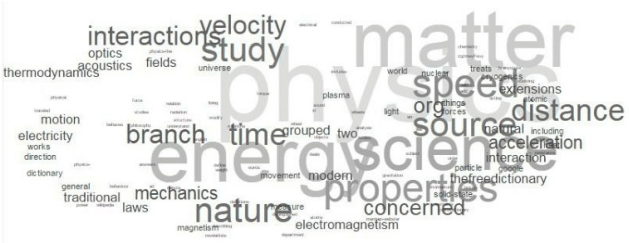
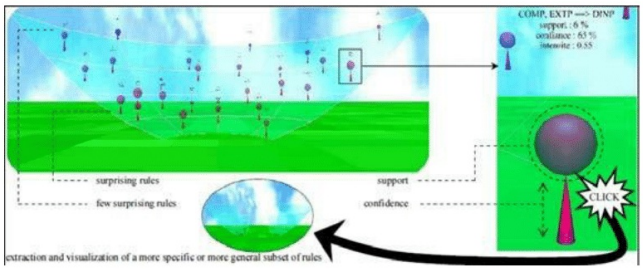
Figure 

Figure 2 [18]: 

A word cloud is made up of six dimensions: width, length, word width, length, color and orientation. The words themselves have statistics about them that come from the various natural language processing that will be applied in this particular application. These NLP stats will drive the value and normalization of the dimensions of the word cloud. Width and length of the page will be kept static as to not imply anything to the user that comes from the data, however changing the words size, color and orientation will allow the user to get the most out of the data present on the screen while not over-whelming them. The CDF or a cumulative statistic of a

word could be used to show changes between specific time periods in regards to one author or site. Doing this with a timeline slider would be intuitive to current mental models a user has and would allow users to mark milestones in the timeline that are relevant to them or to show how an opinion of a site towards a specific topic changes. For example, the alt-right, the libertarian movement, and the socialist party have recently grown roots in the US. If certain news sites are dedicated towards the promotion of these ideals, the settling of these beliefs systems platforms would manifest in some way in the language that those sites use for specific time periods.

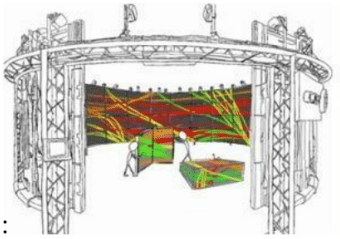
Tying Data to Visuals

Data streams from public social media sites have been used to classify hotels. [25] Applying the same concept to news data would be a way to evaluate the quality of journalism on a site. Yelp reviews from over a thousand users was used to determine the quality of 5-1 star hotels. Since the product of a news site is the content that they publish, being able classify and visualize why your classifier makes its decisions would allow consumers better insight into the bias of their news. Objectivity of the classifier is important which is why it will be purely statistic. Since NLP can be used to generate statistics about words and sentences, it would be informative to use this initial processing step to drive visualization. Figure 3: shows the results of passing the contents of the paper this section was based on through a word map generator. It shows a lot of information about its topic; combining that with more sophisticated NLP libraries would allow researchers to skim articles in less time in order to verify the topic and sentiment of a paper or article. If a reader thinks an article is biased and they could pass the URL to an API that would classify and visualize the language used in an informative way, they would have a better grasp of the high level contents of said article.

Figure 3: 

Interactivity

Interactivity allows a user to discover information that is particularly relevant to a specific user. [22] The overuse of interaction can make a visualization into a game and not be driven to promote discovery. It can also become an expensive and time consuming way for developers to show their skills as in figure 4. There currently is not enough adoption of virtual and augmented reality to display data in a way that it can be consumed. Also, the human brain typically does not respond well to 3D input that is not directly tied to length width and depth and so does not fit the type of data being used in this application. It is much more important for a user to be able to see changes over time in the case of news because populations change the way they think about certain topics and to be able to see that is more valuable than distracting them with extraneous dimensions.

Figure 4 [26]: 

Conclusion

Providing an objective way to show users news data must be available to increase transparency into journalism. Doing this in a dynamic way increases the number of users that could potentially benefit and doing it in a visual fashion makes instruction of analysis minimal. Keeping interaction to a minimum decreases distraction and lets the data speak for itself. Taking all things into account, D3's word cloud is the best method to show a user the reduced dimensionality of articles in browser, a medium that is already widely adopted.

Clustering vs Classification of Newspaper Articles By Topic

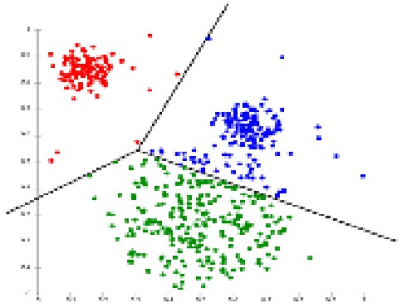
Jared Welch

News Data presents unique challenges compared to numerical data in the realm of machine learning and classification of that data. Due to the nature of the text, reducing the noise of the dataset is also a primary concern. Many words are not valuable. In order to classify the data, this project will aim to first classify by topic, based upon word frequencies in the article. There are many different approaches for estimating the weights of the keywords within articles. The primary considerations when choosing between these techniques include the difficulty in implementation, the overhead of its calculations, and the accuracy of the results. Accuracy in this case means achieving valid topic classification of the articles, which accurately reflect the true contents of the article. For example, if an article is truly about a popular sports team and its players, classification of this article should rank it as related to other articles about the same topic. The problem of topic classification of news data is complex, and there are trade offs depending on which method is chosen. The problem addressed will be deciding upon the most appropriate method to achieve valid topic clustering or classification within the text data.

The first consideration to address is reduction of noise. Before the news data is analyzed, so called ‘noisy words’ such as ‘and’, ‘the’, and other words which can be used in many contexts must be filtered so they do not skew the results of the data. There are various techniques for this, but discussion of stop word removal is outside the scope of this paper. In the case of this project, most of it is written using Python and

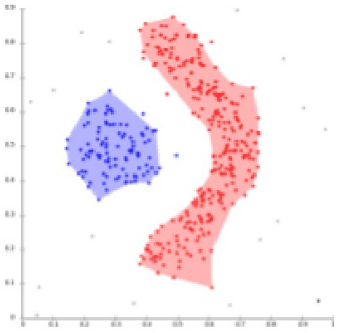
Python libraries. Because of this, removal of stop words will be a pre-processing step before topic classification. There exists an NLTK package which can be used to remove stop words from the text [27]. Due to ease of use, this will be the method chosen to filter out noisy words from the data. Another valuable tool in preprocessing is stemming. Stemming algorithms remove words such as training, trained, and trains can all be replaced by their root train [28]. This also helps reduce the size and feature set of the data before processing further, increasing the efficiency of the classification by reducing the size of the data and removing unnecessary tenses and forms of root words.

The first technique to discuss is clustering. This process, at a high level, involves creating clusters of data, where each cluster represents a grouping of similar articles based upon their keywords[29]. One important feature of clustering data is that the data sorts itself out based upon measures of similarity between the data. There are several different types of clustering. The three main types are connectivity based clustering, centroid-based, and distribution based. Connectivity based clustering is based on the idea that those items more closely related will be closer together in distance. Centroid clustering involves representing clusters by a centroid, meaning a vector which represents the cluster as a whole. Then comparing these vectors, the similarity can be measure by how close in distance the representative centroids are [30]. A popular implementation of this technique is K means. Generally, centroid clustering requires choosing K clusters in



Graphical Representation of K-means clustering [36]

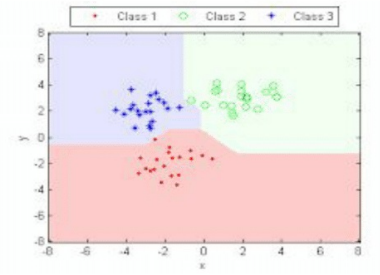
advance, which can be considered one its drawbacks. Finally, another clustering method is Distribution-based, which organizes data points within clusters by their most likely placement within a distribution. Those data points most likely to appear within the same distribution are clustered. This method is generally less suited to text data, and more suited to statistical data, so it does not seem appropriate to the problem at hand. Of these, it appears that a variant of K-means, called bisecting K-means, is most appropriate for text clustering, and performs better than hierarchical clustering (connectivity based)[31]. In light of this, it appears bisecting k-means is the best solution to the problem. However, this technique is unsupervised learning and does not allow for choice of classification, that could be a potential drawback. Supervised learning introduces the element of choice into this process.



Density Based Clustering [35]

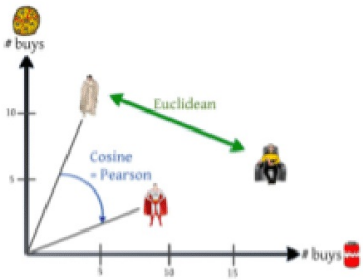
The supervised learning technique most suited to what has been described, is K nearest neighbors. This technique is similar to k-means in that it classifies based upon how similar the data point is to other data points, and place in a class accordingly. The advantage offered with KNN that k-means lacks in this case is that k-means randomly assigns clusters, where as KNN is more used to classify. This classifies the data based upon how similar it is to the training data (or weighted vectors from pre-determined tf-idf values from chosen topic documents) provided initially. Indeed, supervised learning could yield poor results if the initial training data is weak, and clustering does not have

this drawback. Below, a diagram of KNN classified data can be seen. Notice that those in close proximity are considered similar, and groupings of similar data points create class boundaries.



KNN clustering diagram [34]

KNN solutions require a distance function to measure how similar data points are. It is important to vectorize the text for this reason, and represent it in such a way that the vector accurately represents the text within the document, as without a numerical representation of the document, it is impossible to measure difference in a meaningful way to KNN. One such way to represent the data is TF-IDF, which stands for term frequency-inverse document frequency. TF-IDF is a way to measure how important certain words are in a document, and those weights allow representation of the document in a vector format[32]. Using the TF-IDF weights, a similarity score can be assigned using cosine similarity, which measures the angles between the two vectors to determine how similar they are. Another way to measure similarity between vectors is Euclidian distance. Under experimentation, at high dimension it appears that cosine similarity and euclidean distance perform equally well as metrics for similarity[33].



Comparison of Cosine Similarity and Euclidean distance [37]

Because python is the primary language for the project, and cosine similarity can be calculated using python libraries already available, cosine similarity will be the choice of similarity measurement. Indeed, these metrics can also be utilized in the previously mentioned clustering techniques to group the data, grouping those most similar by using these metrics.

In the scope of news data, since it is not clear what topics might be important, it seems less appropriate to use supervised techniques to classify the data. Indeed, since training data will not be readily available, and would take time to generate, it will increase the project overhead quite a bit to gather this training data. In light of this, it appears unsupervised clustering using k-means is the optimal solution for this problem. This will allow the data itself to decide upon which clusters are most important, and a subset of these clusters can be chosen as representing certain topics, based upon the keywords contained.

An important consideration in choosing K-means as a technique for clustering the data is that there are already readily available implementations of this algorithm, and it is relatively simple to implement with good accuracy of the results of the data. A well defined algorithm and procedure with existing implementation is important for two reasons. First, it allows for more streamlined development, as it is more manageable to develop software that implements existing techniques. Since time is a factor in solving this problem, it is important to implement a solution that is already proven and functional instead of trying to create the software from scratch based on theoretical concepts. And second, using techniques that have been implemented many times already suggests

precedent and validation for the technique's usefulness. Less tested techniques might not yield as accurate results, or be as straight forward to implement, and k-means overcomes both of those parts of the problem.

Addressing the problem of topic classification in text documents is complex and many considerations must be made. First, it is important to reduce the noise of the dataset in order to prevent bias from the many placeholder words such as ‘the’. Once the dataset has been reduced, using NLTK libraries readily available, classification can begin. Depending on TF-IDF similarity of the document to the training data, KNN classification will place the document in proximity to its closest neighbors, where closest means cosine similarity value that is closest to 1. This classification has precedent, is relatively simple to implement compared to the other techniques, and is an accurate measure at higher levels of dimensionality, which will be achieved with ease due to the multidimensional nature of text. Unsupervised clustering techniques fit all the criteria necessary, but do not allow for classification to compare the data against; rather clustering removes the ability to choose the topics themselves, but in this case that is actually more desired, so that training data is not needed to gather results. K-means clustering will allow classification in later stages based upon the data itself, rather than decisions made by developers in advance. This should result in clusters that are chosen based upon their significance, rather than chosen in advance, which should overall increase the accuracy of the project’s classification of the data at later stages.

Works Cited

[1] Kalyani, Joshi, Prof Bharathi, and Prof Jyothi. "Stock trend prediction using news sentiment analysis." arXiv preprint arXiv:1607.01958 (2016).

[2] Lutz, Ashley. "These 6 Corporations Control 90% Of The Media In America." Business Insider. Business Insider, 14 June 2012. Web. 20 Apr. 2017.

[3] Ramdass, Dennis, and Shreyes Seshasai. Document Classification for Newspaper Articles. MIT, 18 May 2009. Web. 23 Apr. 2017.

[4] Puget, Jean Francois. "The Most Popular Language For Machine Learning Is ... ." IBMDeveloperWorks. N.p., 19 Dec. 2016. Web. 17 Apr. 2017. <https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Language_Is_Best_For_Machine_Learning_And_Data_Science?lang=en>.

[5] Binga. "Python vs R for machine learning." Data Science Stack Exchange. N.p., 3 Jan. 2017.Web. 17 Apr. 2017. <https://datascience.stackexchange.com/a/339>.

[6] Alvas. "Java or Python for Natural Language Processing." Stack Overflow. N.p., 12 Apr. 2017. Web. 17 Apr. 2017. <http://stackoverflow.com/a/22905260>.

[7] Markham, Kevin. "Should you teach Python or R for data science?" Data School. Data School, 18 Apr. 2016. Web. 17 Apr. 2017. <http://www.dataschool.io/python-or-r-for-data-science/>.

[8] "The Computer LanguageBenchmarks Game." Python 3 vs Java. N.p., n.d. Web. 17 Apr. 2017. <http://benchmarksgame.alioth.debian.org/u64q/python.html>.

[9] Wickham, Hadley. "Performance." Performance · Advanced R. N.p., n.d. Web. 17 Apr. 2017. <http://adv-r.had.co.nz/Performance.html>.

[10] Simmering, Jacob. "How slow is R really?" R-bloggers. N.p., 28 Jan. 2013. Web. 17 Apr. 2017. <https://www.r-bloggers.com/how-slow-is-r-really/ >.

[11] Mittal, Chandan. “Which is better for web applications, Java or Python?” Quora. Web. 18 Nov. 2017. <https://www.quora.com/Which-is-better-for-web-applications-Java-or-Python>.

[12] "Programming languages." Programming Languages | Hammer Principle. N.p., n.d. Web. 17 Apr. 2017. <http://hammerprinciple.com/therighttool>.

[13] Munroe, Randall. Xkcd: Python. N.p., n.d. Web. 17 Apr. 2017. <https://xkcd.com/353/ >.

[14] Kiser, Matt. "Introduction to Natural Language Processing (NLP) 2016."*Algorithmia*. N.p., 11 Aug. 2016. Web. 19 Apr. 2017. <http://blog.algorithmia.com/introduction- natural-language-processing-nlp/>.

[15] "Natural Language Toolkit."*Wikipedia*. Wikimedia Foundation, 23 Apr. 2017. Web. 19 Apr. 2017. <https://en.wikipedia.org/wiki/Natural_Language_Toolkit>.

[16] Robert. "NLTK vs. spaCy: Natural Language Processing in Python."*The Data Incubator*. N.p.,27 Apr. 2016.Web.19Apr.2017. <http://blog.thedataincubator.com/2016/04/nltk-vs-spacy-natural-language-processing-in-python/>.

[17] *Stemming 1.0*. N.p., n.d. Web. 19 Apr. 2017. <https://pypi.python.org/pypi/stemming/1.0>.

[18] Blanchard, J., Guillet, F., & Briand, H. (2003). A user-driven and quality-oriented visualization for mining association rules. *Third IEEE International Conference on Data Mining*. https://doi.org/10.1109/ICDM.2003.1250960

[19] Byrne, L., Angus, D., & Wiles, J. (2016). Acquired Codes of Meaning in Data Visualization and Infographics: Beyond Perceptual Primitives. *IEEE Transactions on Visualization and Computer Graphics*. https://doi.org/10.1109/TVCG.2015.2467321

[20] Chen, L., & Zhou, H. (2016). Research and application of dynamic and interactive data visualization based on D3. *2016 International Conference on Audio, Language and Image Processing (ICALIP)*. https://doi.org/10.1109/ICALIP.2016.7846608

[21] Fu, T. c., Sze, D. C. M., Leung, P. K. C., Hung, K. y., & Chung, F. l. (2007). Analysis and Visualization of Time Series Data from Consumer-Generated Media and News Archives. *2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*. https://doi.org/10.1109/WI-IATW.2007.104

[22] Jacucci, G. (2016). Resourceful interaction in information discovery. *2016 IEEE 32nd International Conference on Data Engineering Workshops (ICDEW)*. https://doi.org/10.1109/ICDEW.2016.7495639

[23] Naidoo, J., & Campbell, K. (2016). Extended abstract: Best practices for data visualization.

2016 IEEE International Professional Communication Conference (IPCC). https://doi.org/10.1109/IPCC.2016.77405097

[24] Setlur, V., & Stone, M. C. (2016). A Linguistic Approach to Categorical Color Assignment for Data Visualization. *IEEE Transactions on Visualization and Computer Graphics*. https://doi.org/10.1109/TVCG.2015.2467471

[25] Suzuki, T., Gemba, K., & Aoyama, A. (2013). Hotel classification visualization using natural language processing of user reviews. *2013 IEEE International Conference on Industrial Engineering and Engineering Management*. https://doi.org/10.1109/IEEM.2013.6962540

[26] Thomas, B. H., Marner, M., Smith, R. T., Elsayed, N. A. M., Itzstein, S. Von, Klein, K., … Suthers, T. (2014). Spatial augmented reality — A tool for 3D data visualization. *2014 IEEE VIS International Workshop on 3DVis (3DVis)*. https://doi.org/10.1109/3DVis.2014.7160099

[27] Information about NLTK python package for removing non-essential words from text https://pythonspot.com/en/nltk-stop-words/

[28] Discussion of Machine Learning Techniques applied to Text http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.9153&rep=rep1&type=pdf

[29] Book detailing clustering techniques Cluster analysis, 5th edition

Brian S. Everitt, Sabine Landau, Morven Leese, Daniel Stahl

February 2011, ©2010

[30] Stanford resource for clustering techniques and relevant implementation details https://web.stanford.edu/class/cs345a/slides/12-clustering.pdf

[31] Discussion of clustering in relation to documents and text http://glaros.dtc.umn.edu/gkhome/fetch/papers/docclusterKDDTMW00.pdf

[32] Breakdown of K means clustering with text data https://www.codeproject.com/Articles/439890/Text-Documents-Clustering-using-K-Mean s-Algorithm

[33] Discussion of cosine similarity and euclidean distance in KNN http://www.cse.msu.edu/~pramanik/research/papers/2003Papers/sac04.pdf

[34] Image example of KNN http://www.peteryu.ca/tutorials/matlab/visualize_decision_boundaries

[35] Image of Density based clusters https://en.wikipedia.org/wiki/Cluster_analysis#/media/File:DBSCAN-density-data.svg

[36] Image of Kmeans

https://en.wikipedia.org/wiki/File:KMeans-Gaussian-data.svg

[37] Image of Euclidean Distance

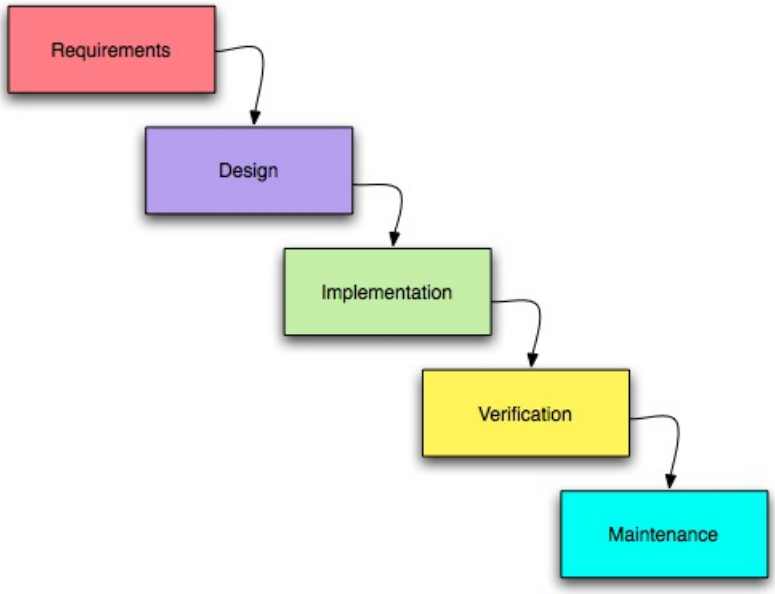
https://comsysto.wordpress.com/2013/04/03/background-of-collaborative-filtering-with-m ahout/

Development Plan

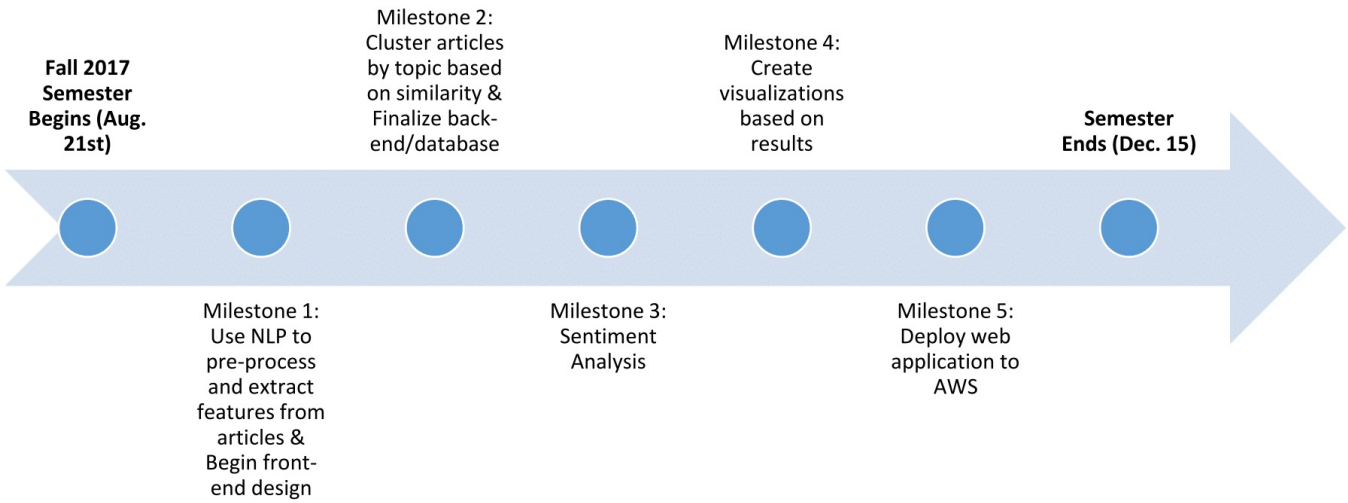
Development Method

Waterfall methodology follows the following steps: First capture all of the requirements of a software system, which has been done via this documentation. Next, start analyzing the scraped data using the NLP and machine learning python libraries and creation of the data warehouse. Design will be next so the layout of the website can be solidified and work will begin on the visualization. Coding will be done throughout all of these steps, along with testing; however a testing suite will be implemented last. And finally the software will be deployed from DEV to PROD, ending Capstone II and 2017 Seniors at Mizzou. Any further support, migration or maintenance will come through all members of the team for review and verification. The main reason waterfall is best for this project is because many of the methodologies that could potentially be used to attempt a task may fail, but that developer should still succeed at their task. Research is often done this way since it is difficult to partition

out the different smaller goals. For example, having someone do the clustering piece or be in charge of it is better than doing it in smaller chunks. This is because they could while researching and prototyping discover some new way to do clustering that invalidates their work on that problem up to this point. They should continue chugging away at this new idea. If agile was used for this project someone would have a week to implement X clustering algorithm. If it works awesome, if not, they've failed and now the timeline is wrong, since time is not built into this agile development for failure and research since its done at such a granular approach.



Timeline



Week Number	Date Range	Agenda
1	August 21 - August 27	Start Milestone 1: Use NLP to pre-process and extract features from articles & Begin front-end design
2	August 28 - September 3	Testing & Meet as a group
3	September 4 - September 10	Meet as a group
4	September 11 - September 17	End Milestone 1, Start Milestone 2: Cluster articles by topic based on similarity & Finalize back-end design
5	September 18 - September 24	Meet as a group
6	September 25 - October 1	Meet as a group
7	October 2 - October 8	Meet as a group
8	October 9 - October 15	End Milestone 2, Start Milestone 3: Sentiment Analysis
9	October 16 - October 22	Meet as a group
10	October 23 - October 29	Meet as a group
11	October 30 - November 5	Meet as a group
12	November 6 - November 12	End Milestone 3, Start Milestone 4: Create visualizations based on results & Deploy web application to AWS
13	November 13 - November 18	Stretch goals? Meet as a group
14	November 27 - December 3	Meet as a group
15	December 4 - December 10	End Milestone 4
16	December 11 - December 15	Finishing Touches

Work Delegation

- Kurt: Data-warehousing/ data visualization
  - Data warehouse development to create data marts that ease reporting analytics
  - Create visualization based upon categorization and statistics done via analysis using Javascript (D3)
- Ali: Backend web development
  - Develop a normalized transactional database using PostgreSQL
  - Backend web development using Python (Flask)
- Justin: Natural Language Processing
  - Natural language processing for cleaning data and extracting features for use in machine learning analysis
  - Development of web scraper
  - Design of AWS deployment prototype
- Jared: Machine Learning Analysis
  - Categorization of articles using both unsupervised and supervised learning techniques
  - Web server deployment
- Zach: Front End Development
  - UI design using HTML, CSS, and JavaScript
  - System administration

Possible Troubles

- In general, this is a big, challenging project for undergraduates who will be required to learn many new techniques to bring it to the finish line.
- Using AWS offers great potential for decoupled, scalable data processing but this could be limited by running up against financial constraints.
- It might be difficult to get a high degree of accuracy in this application of unsupervised clustering due to the nature of news articles and their tendency to contain overlapping topics.
- It might prove difficult to draw accurate conclusions with sentiment analysis due to the typically neutral language used in most news articles.
- If any of the data needs to be labeled by hand, this could prove to be incredibly time consuming and, due to the small size of the development team, would be vulnerable to bias.

Alternative Strategies

- If AWS proves to be unfeasible for the large amount of data processing required due to financial constraints, the team has access to the university research cluster computer to process data locally, outside of the cloud and its associated costs.
- If unsupervised clustering proves inadequate for topic classification, we can fall back on supervised learning using K nearest neighbors or other supervised learning algorithms on labeled data.

Testing

We plan on testing using unit testing, creating testing suites with each new version release of the software. As new features are added, unit testing will allow us to continuously ensure old functionality while introducing new features. There remains research to be done in this area as far as testing software to use. However, a general plan for testing will resemble the following:

- As new features are written, verify their functionality with unit tests, ensuring all tests are passing before a feature is finished. Alternatively, development may sometimes begin with a test first style of development, writing code to match the tests perceived to be important to that feature.
- Tests will be consolidated in one testing suite, which will allow old tests to be run along with new features to test that the new features do not hurt existing functionality.
- A stretch goal for this project will be to implement testing hooks and build automation so that all live code complies with testing before being put into a live environment. This ensures highly stable code releases, and prevents bad code from being released.
- Database Integration Testing Since our data is what drives this projects, we need to be aware of its status. To do this, I will create a bar chart that gives a count of all null fields in each column of the database and check it by weekly. If it seems like the count is getting too high we will need to check our scraper to confirm it is passing its unit tests and that it's passing information to the database the way we intended it too

Below is a general outline of what we think will need to be tested as we develop.

Feature	How we will need to test
TF-IDF keyword values	For this we can test our TF-IDF by giving articles with known TF-IDF values and testing the accuracy of our TF-IDF methods
Clustering	We will need to verify the accuracy of grouped clusters. We will do this by comparing word frequency of articles pulled from a cluster and see if those articles actually share keywords
Topic Classification	We will test this by sampling a few random articles from each topic cluster after we train it and verify the articles relate to that topic
Sentiment Analysis	To test this, we will need to compare sentiment analysis results with the actual content of the article and judge whether the sentiment is expressed accurately
Database stored procedure testing	In the general case, we should have test data to run stored procedures on and test the results to verify they work as intended
Stop word Removal	Similar to TF-IDF, we will need to verify our stop words are being removed by giving sample articles and checking the output for remaining stop words
Stemming/Lemmatization	Given sample words, test the expected stem output with process
Data warehouse from current database	We will need to verify when we create our data warehouse that the data accurately reflects the database data; this can be done by comparing queries and their output from the database vs the data warehouse
Flask/Front End testing	The best way to test front in features seems to be with integration tests and workflow procedures to verify that the old workflows work as new features are added. This does not yield itself well to unit testing
Web scraper	We can test the scraper is working by checking for recently posted articles at sites we expect to scrape properly and verifying those articles are contained in the database as expected
Visualizations	Testing visualizations should be relatively simple using dummy data to visualize and verifying it is displayed accurately

Prezi

- http://prezi.com/ybpo3byyhwaw/?utm_campaign=share&utm_medium=copy