DVA PROJECT FINAL REPORT
IDENTIFYING AND VISUALIZING HISTORIC NEWS TRENDS
GROUP NUMBER: 13

INTRODUCTION

Anyone interested in learning about historical events would also be interested in a one-stop destination for visualizing historical news trends. Unfortunately, popular news services like Google Trends lists the trending topics only for a limited time period. We intend to bridge in this limitation by using the New York Times news archives containing articles dating back to the early 20th century.

PROBLEM DEFINITION

The goal of our project is to identify and visualize historical news trends for the past 100 years within a time period (week, month or year) selected by the user.

SURVEY

In our literature survey, we identified a number of papers on modelling news activity. However, none of them was specifically focused on visualizing the trending news topics in the past. We have categorized the related work into the following sections:

Retrospective event detection in a single stream

Early research efforts on topic detection have focused on identifying events retrospectively, ie, by looking back into data. Topic Detection and Tracking [1] is one of the pioneer research projects on segregating news stream into articles and on clustering the related news articles into topics. [2] uses Group Average Clustering - a hierarchical clustering algorithm for detecting events. It uses lexical similarity and temporal proximity for clustering events. [3] proposes BurstVSM, a model that incorporates temporal information for article clustering and event detection. [4] solves the event detection problem by splitting the article into categories like who (*Persons*), when (*Time*) etc. and then running probabilistic models on each category to identify the associated event. Since our project also deals with detecting trending news topics retrospectively, we intend to explore these approaches in our project.

Live new event detection in a single stream

[2] uses incremental clustering to identify new events in streams on the fly. [6] extends the incremental TF-IDF model by normalizing the similarity scores based on the source, topic and rules of interpretation. [7] proposes a multi-stage vector space algorithm based on topic categorization and named entity overlaps for detecting new events. [8] differs from [7] in that it uses a dynamic clustering method for storing similar events in a

hierarchical index tree instead of using predefined categories. [9] and [10] identify how trends emerge, survive and eventually fade away over time. It uses techniques like KNN and first-order differential equation in identifying popular trends. Approaches discussed in [7] and [8] are relevant to our project since an efficient clustering algorithm is required to provide a superior user experience with minimum delay between user queries.

Identifying events in multiple streams

A number of research efforts have focused on discovering named entities with simultaneous bursts in multiple news streams. [5] proposes a probabilistic algorithm to mine bursty topics from multiple news streams. [11] presents a formal approach for modelling bursts of activity in document streams while [12] uses variants of Linear Discriminant Analysis in identifying topics from correlated news articles and text streams. [13] uses Markov Modulated Poisson Process probabilistic model for normalization across multilingual news streams. It also uses a dynamic programming approach to deal with the time gaps between the bursts. Although the scope of our current project is limited to identifying the trending topics from a single news stream, using multiple streams would be an interesting extension to our project.

Generating timeline overview

Some of the research around modelling news activity has been on generating a timeline for the news topics. [14] deals with the problem of identifying breaking points in a news topic, given a set of articles on that topic. To detect a breakpoint, it uses HMM models to analyze theme variation before and after a point. [15] uses a statistical algorithm to analyze text in the article and extract important topics and automatically generate timeline and labels. These works are related to our project because we display the relevant news in chronological order when the users click on a specific topic.

PROPOSED METHOD

Intuition

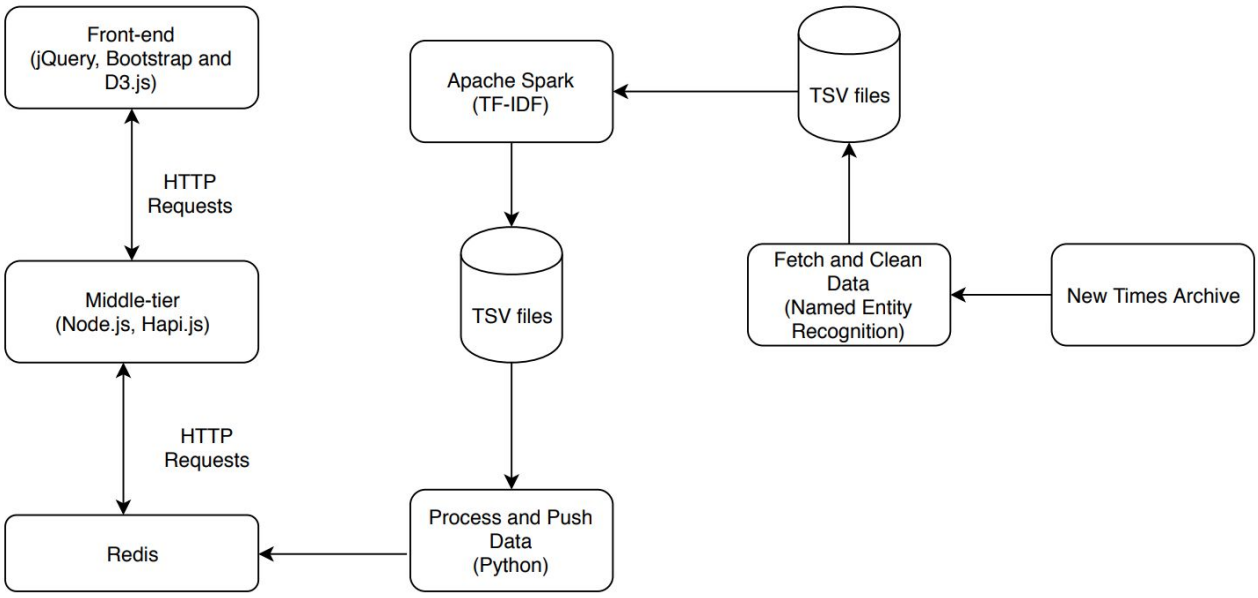
The intuition behind our approach is that popular topics tend to show up multiple times in news articles and the frequency of appearance of the topic in news over a time period is a reasonable estimate of the popularity of the topic. We used Named Entity Recognition algorithm to group the entities in the news articles since popular news topics tend to frequently revolve around entities. We used the TF-IDF algorithm to assign weights to individual words in the document and filtered out only those words where the TF-IDF score is above 10. However, we didn't use the weights assigned by TF-IDF for ranking the topics since TF-IDF can only find the words characteristic of each news article and hence assigns lower weights to topics that frequently appear across articles. So, we counted the number of appearances of the topics by considering

only the TF-IDF filtered words in each news article. The top 10 identified topics over the selected time period are then visualized by our front-end web application.

Our approach is better than the state of the art because to the best of our knowledge, there is no existing tool that allows us to view the trending topics by month/year for as long as 100 years in the past and also visualize the news articles related to each topic. For example, Google Trends shows the trending topics for each year only starting from the year 2006 and doesn't have an option to search for trends by month. Plus it relies on user searches and not news articles to identify the trending topics. Wikipedia, another popular tool for searching popular events, lists specific events that happened on specific dates, but doesn't have a way of visualizing the trends and related news. Therefore, the user can lose track of the bigger picture.

Description of approaches

Block Diagram:



Data Processing:

The first step of the process is downloading data from NYTimes Archive API. It has data from 09/1851 to present (except 09/1978 and 10/1978). On sending the HTTP request, the response is a list of all NYTimes articles for the requested month (only headline, abstract, lead paragraph, date, keywords and pictures, no internal content), in JSON format. This was converted to tab separated values for easier integration with Apache Spark which is used in a later stage of the data processing pipeline.

The second step involves *Data Cleaning* and *Stop-Word Removal* using python scripts. We removed news articles having missing news headlines and content. We observed that punctuations created inconsistencies in the TF-IDF algorithm. For example, a dot('.') present at the end of a sentence resulted in multiple keywords representing the same word. *Named Entity Recognition* was also used for grouping together entities having multiple words, thereby ensuring that the entities are having a single unique representation.

The third step is *TF-IDF Based Clustering* for identifying relevant keywords in the document. TF-IDF assigns weights to words taking into account word frequency within a document and in the entire corpus. TF-IDF calculation was carried out on Apache Spark. Instead of selecting top 10 keywords with high scores, repeated keywords having either a score more than the threshold value or the keywords which were part of the original NYTimes article were grouped and counted together. After the aggregation, keywords with top 10 counts were selected.

Data Interface:
We aggregate the popular keywords for each year and month. A batch job transforms the csv output from the Apache Spark instance into key values pairs and stores it in a Redis instance. This information is retrieved by the Node.js Web application framework for rendering the visualization.

Visualization:
We used hapi.js framework and Node.js in the backend of our web application. We exposed web services that connects with the Redis database for fetching the popular news topics by year and/or by month, and for retrieving the news articles related to a given trending topic via the hapi server. When aggregating news articles related to a topic, any article that comes with an image is ranked higher than articles without images. The top 10 related news articles based on this ordering is then exposed as a REST API.

We used D3.js, jQuery and Bootstrap CSS framework for creating the user interface. We explored several visualization techniques like bar charts, bubble charts and line charts to effectively visualize trending topics and found that bubble chart [Figure 1] creates a visually appealing interface without overwhelming the user. The user can select the time period over which he/she wants to explore the news trends. When the user selects the time frame and starts the visualization, the application makes REST requests to the Node.js server for fetching the trending topics in that time period. A transition of the trending topics is then displayed from the start period to the end period

in steps of one month each. The user can pause the visualization at any point of time and adjust the slider manually to skip to a different time frame.

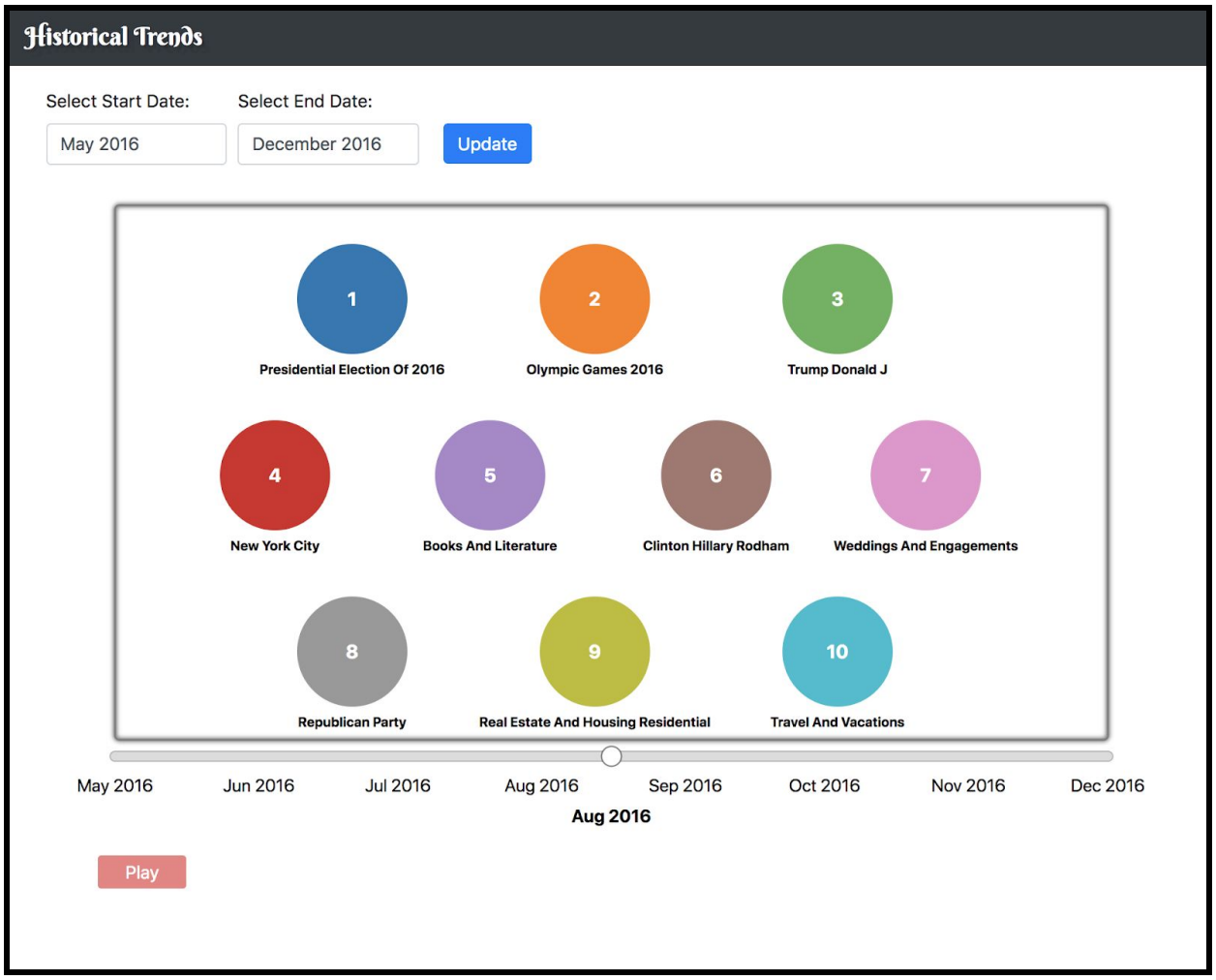


Figure 1

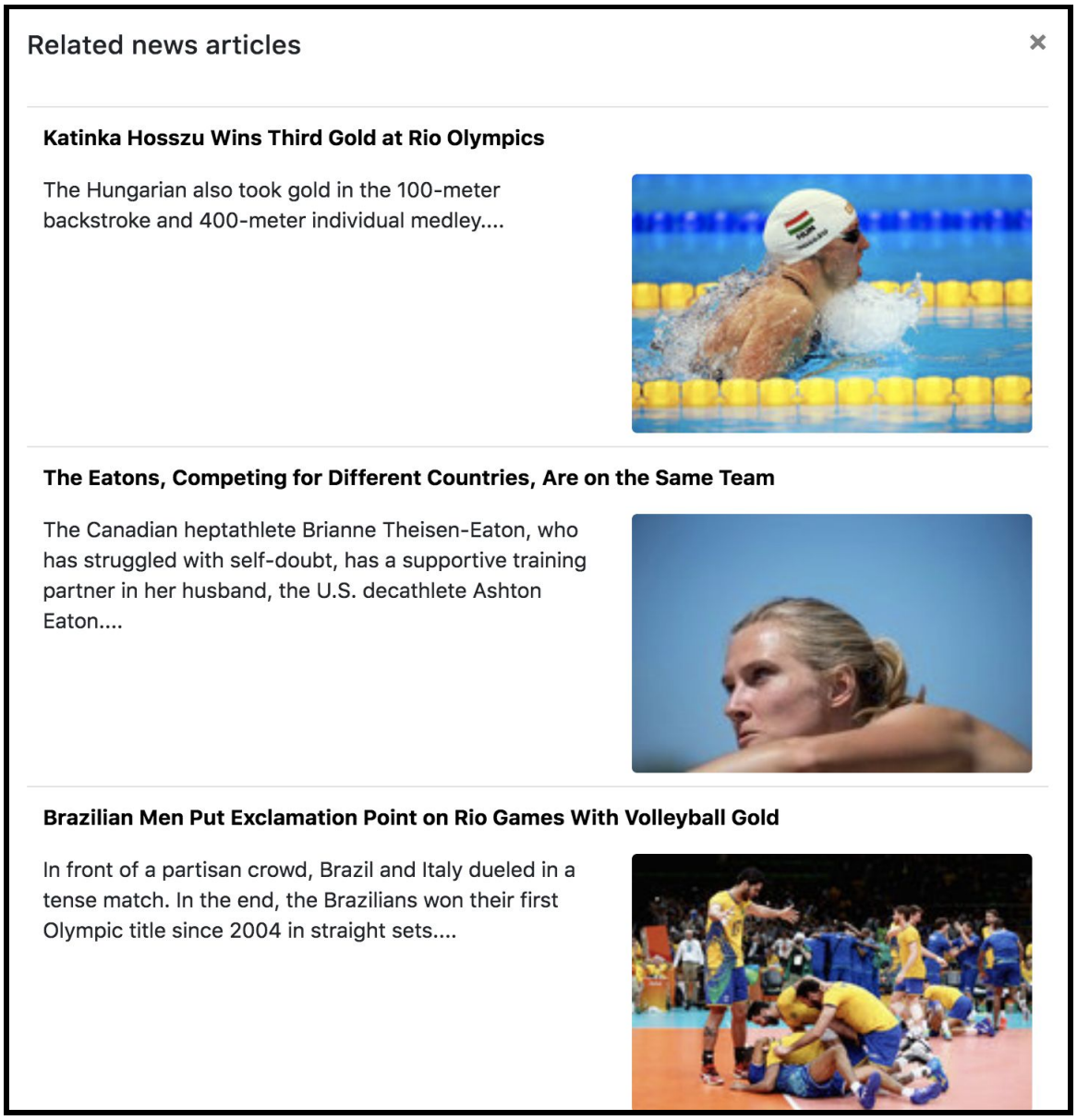


Figure 2

The user can click on a topic after pausing the visualization to view all the new articles [Figure 2] related to the selected topic in the specified time period. The application then makes web service requests to the backend to fetch this data. The user can browse through brief snippets on the articles including the headlines and images and can view more details on the news topic by clicking on the hyperlinks that takes them to the New York Times news archives. The user can switch back to the time series of trending topics by closing the modal and continue the visualization.

EXPERIMENTS/EVALUATION

Testbed

Testing is done on a standard computer with the web application loaded on a modern web browser (Chrome/Firefox/Safari etc.) with javascript enabled. We installed a Redis database on a Windows server and loaded the database with the trending topics and related news articles output by our algorithm. We ran the Node.js application locally.
Following are the objectives of testing:

1. How much time each does algorithms take to processes the data?
2. Do the algorithms identify trending topics accurately?
3. Is the Graphical interface user-friendly and self-explanatory?
4. How do the news trends behave? i.e. In what manner does the ranking of news topics change over time?

Experiments/Observations

1. Performance of algorithms:

- We tried using Word2Vec for mapping a keyword to vector space and then grouping them into topics using k means clustering algorithm but the mapping of words to vector space did not turn out useful for our specific application. We also tried out topic summarization using embedded keywords for identifying trending topics in a news document. Since both approaches did not yield satisfactory results, we used a TF-IDF based approach in our algorithm.

- Data aggregation scripts were initially implemented using the Pandas python library. The script was performing well on a single file and the performance degraded as more files were created by the TF-IDF scoring algorithm. The performance degrade was due to the fact that Pandas data-frames are immutable and multiple copies of the data were created while trying to append more results.

- We observed that the download script takes 10 seconds per month to fetch and clean the data, 1 to 2 minutes per month to run the TF-IDF algorithm and store the processed data in Redis instance.

- Since the data is already stored and available, the front end application can display user requested specific topics without any noticeable delay.

2. Evaluating Accuracy:
We evaluated our approach by comparing the result of our algorithm against the event data from Wikipedia pages. As the NY times gives more importance to US news, we used events in Wiki US news as well as Wiki world news to check the accuracy of the application.

Top 20 keywords for some randomly selected months in the period of 2014-2017 were compared with event data from the Wikipedia page entries for those years. (e.g. “2017 in the United States”). The accuracy was calculated as the percentage of keywords in our top 20 results that have a matching event on the Wikipedia page. We got a net accuracy of 69% in our evaluation.

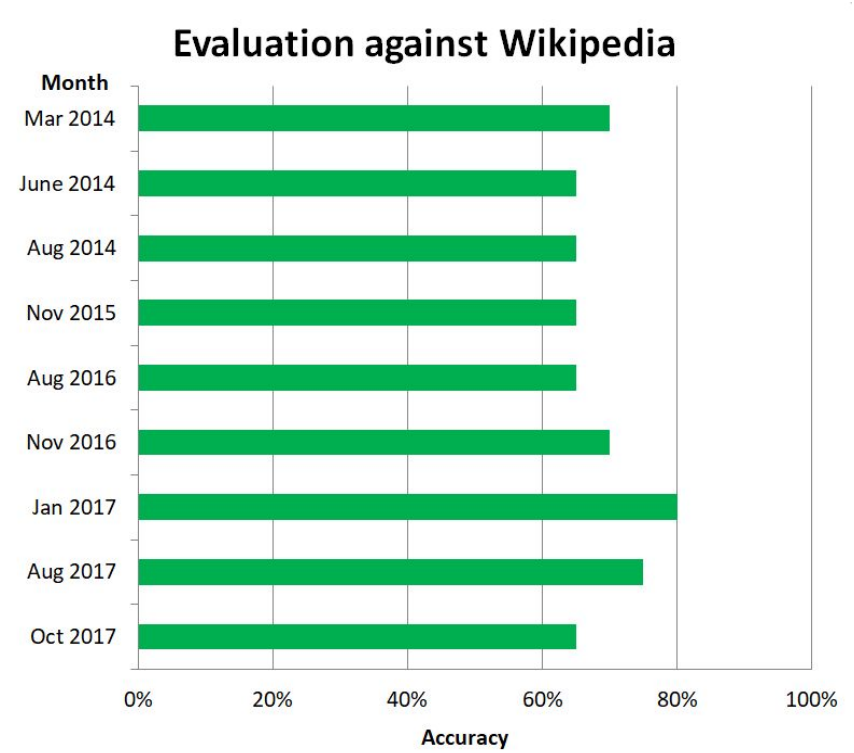


Figure 3

The limited accuracy of our system is explained by the following points:

- Articles related to politics dominate the New York Times, so smaller categories like entertainment and sports take a backseat.
- New York Times publishes more articles local to New York City which is not reflected in Wikipedia
- Repeated appearance of generic keywords like “Books and Literature”, “Real Estate and Housing” etc. that do not correspond to real-world events

3. User Experience:
We observed 4 users to use our application and we noted their feedback. They had the following suggestions:

- We initially had a design where the topic bubbles appear at random position on the canvas to give a rich user experience. However, users found this to be confusing and overwhelming. The users suggested to fix the position of the topic bubbles, which removed a lot of randomness from the visualization and thus improving the user experience.
- Users pointed out that Figure 4 representation of bubble charts led to empty spaces in the canvas negatively affecting the aesthetics of the visualization.

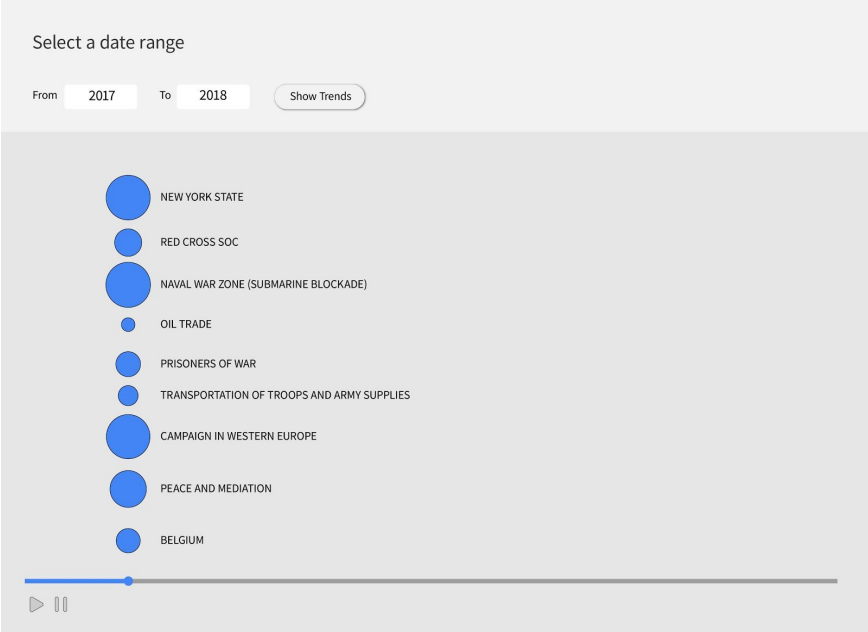


Figure 4

- We also observed that having the size of the bubble represent the topic's popularity resulted in text overflow leading to a poor visualization.

4. Observed behaviour of trends over time:
When the top 20 keywords are observed across a range of consecutive months, the following observations were made:

1. Some keywords were consistently ranked in the top 20 every month
2. Some keywords burst into the top 20 and then faded away quickly
3. Some keywords showed long term increase or decrease over time

These observations can be demonstrated in the form of a line chart. The below data is for the period from April to October 1996.

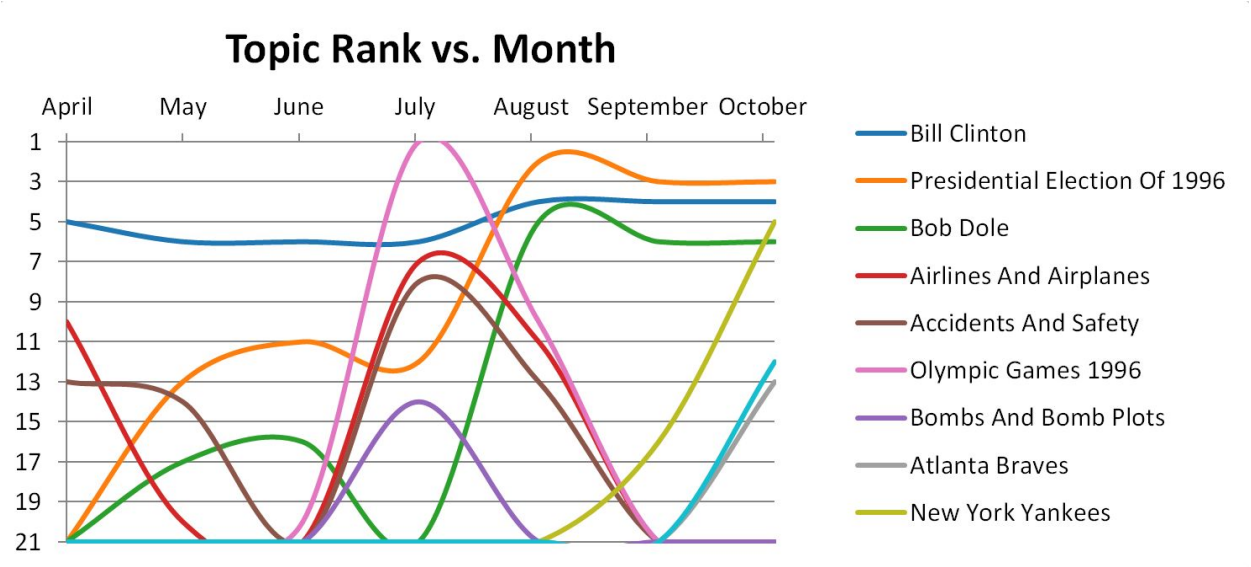


Figure 5

The observations can be explained as follows:

1. Since New York Times mainly covers U.S. politics, the keyword “Bill Clinton”, the incumbent president at the time, appears consistently in the top 20.
2. For that matter, since a change of power is a very important event for New York Times, the hype for the Presidential Election of 2016 starts as early as May and grows gradually as the election approaches in November.
3. Other “less important” events like the 1996 Summer Olympics are given importance only when the event occurs and then they die out quickly.
4. It is also observed that for unexpected events like flight crashes and bombings, the “attack” is faster, meaning the related keywords burst suddenly into the top 20 only after the event occurs. However for planned events, like Presidential Election or Olympic Games, the attack is slower, meaning the burst is not so sudden because these events are discussed even before they happen.

The full list of events in this period is shown in the figure below:

Date	**Event**
April 3	1996 Croatia USAF CT-43 crash
April 11	Jessica Dubroff crash
May 11	ValuJet Flight 592 crash
July 17	TWA Flight 800 crash
July 19- Aug 4	1996 Summer Olympics
July 27	Centennial Olympic Park bombing
October 20-26	1996 Baseball World Series
November 5	1996 U.S. Presidential Election

CONCLUSIONS AND DISCUSSION

We achieved an accuracy of around 70% for identifying the trending topics in a given time period. One of the major reasons hampering our accuracy is that NYTimes has a higher coverage of political and financial news than entertainment and sports related news and that it majorly covers news in the US. One potential solution to address this problem is gathering news data from multiple news sources and then identifying the trending topics from all those sources. This would also help us eliminate the bias introduced by news agencies focussing on specific news topics with political and financial agenda.

As a simple extension to our application, we can automate the process of gathering data from New York Times archives and storing the trending topics by having a script that periodically checks for updates to the New York Times news archives. The rest of the system can then use the newly added data to display the trending topics for the updated time period.

DISTRIBUTION OF TEAM EFFORT

Everyone has contributed equally to the project.