Final Project Report

Introduction:

In this project, I have created a web application which displays the top stories from New York Times in real-time. In addition to the article text, the application also offers insights on the text, like- Text Summary, Key Words (Entities), Positivity Score, Avg. Read Time, Avg. Word length, Avg. Sentence Length.

App URL: https://nyt-inshorts.uc.r.appspot.com/home?section=Automobiles

Data Source:

NYT offers several APIs (https://developer.nytimes.com/apis) to get real-time data on articles, archives, books, movie reviews, etc. on its website. All we need is a developer account to access the APIs. The APIs are free to access with a standard limitation on the number of requests to their server in a month.

For my app, I fetch use the Top Stories API. The API allows access to Top Stories in sections like World News, Business, New York news, Health, Movies, etc.

Functionalities:

There is a navigation bar which enables user to navigate across sections. When the app loads user is prompted with world news articles. Similarly, on click of a section, the user is prompted with articles for that section in NYT. Since, the data is real-time, we can crosscheck our results from the NYT website.

When a user clicks on any article, the article text with some additional analytical insights is displayed. The user also has the option to view the article on NYT website by clicking on the Title.

The analytical insights offered to the user are:

1) Article Summary – an auto-generated summary of the article text. I have used a python library called newspaper3k to obtain the summary from the text. The library uses extractive text summarization techniques to generate the summary.
2) Article Keywords – auto-identified keywords or entities in the article text. The newspaper3k library also generates the keywords in the text.
3) Average Metrics
    a. Average word length – Used the NLTK library in python to tokenize words in the text, and then calculated the average length of words.
    b. Average sentence length- Used the NLTK library in python to tokenize sentences in the text, and further tokenized each sentence into words to calculate the avg. sentence length.

c. Average read time of the article – medium.com has noted that the avg. read time for articles in English is 265 words for minute. This metric is based on their analytical research. I have used this metric to calculate the avg read time of the article.

4) Positivity Score- Strength of positive sentiment in the article. It is measure on a scale of 5. Anything above 2.5 is considered positive. So, a score of 2.5 indicates neutral sentiment. The positivity score is obtained from the analyze_sentiment function in the Google Cloud Natural Language API.

Design:

I have created the application using the flask framework in python. The frame uses a template rendering engine called Jinja2 for rendering the webpages. Each endpoint is tied to a webpage which gets rendered when the endpoint is invoked. Following the list of endpoints in the application:

i) /home: Route to the home page of the application. As mentioned, it displays the Top Stories in the World News section. This endpoint gets called for each selection of a section.

ii) /article: Route to view the article text and insights. The user is routed here on click of the article title in the home page. The user also has the option to view the article on NYT by clicking the article title on this page. The article text and the analytical insights mentioned before are displayed on this page.

APIs: For calculating the positivity score, I have used the Google Natural language API. The analyze_sentiment function of the API provides a comprehensive analysis of sentiment of the overall documents and at the sentence level. In the webapp, I only display the score of the overall document.

Caching: Calculating the insights takes time, particularly the positivity score. So, I have created a Redis instance in cloud Memory Store and used it as a cache for analytical insights. This significantly increased the response for repeated requests.

App Engine: I have deployed the app on Google App Engine.

Future Work: There is a lot of future potential for this app. First off, I could include more data sources from NYT, like movie reviews, NYT best sellers, Archives etc. Also, I could add data from other news companies like Bloomberg, Guardian, Google News, etc. which offer free access to their data APIs. Additionally, I could use other functions in the Cloud Natural Language API for a more comprehensive analysis of sentiment, like Entity Sentiment, Entity Analysis, etc. One other feature would be to add subscription for daily newsletters, this would require the need for a user database which can easily be provisioned in Google Cloud.

Cost:

1) The caching server, a memory store instance, is charged $35.77/month.
2) App Engine instances are charged $0.05/hr.
3) The Cloud Natural Language API is charged $5.00 per predicting 1,000 text records.

References:

1) Google Cloud API docs
2) medium.com
3) https://developer.nytimes.com/apis
4) https://newspaper.readthedocs.io/en/latest/