The Fastest Way to be Informed

I. Functionality requirements

Our goal in this project is to effectively crawl any popular web content and be able to use the data to create a platform that will display in real time the frequent trends in the data we have collected. The collection will be on a daily basis because we want our system to have fresh information. In other words, we want to focus on the regularity of the information we display on our platform. Currently, the amount of data we produce has dramatically increased. Therefore, it is complex to keep track of valuable information that is beneficial and informative. Our system will be designed to give an efficient information pattern to customers that may visit our platform. The information could be related to:

- Sports
- Entertainment

Structure of the Program

The program will contain 5 major subsystems:

1) Web Crawler
2) Web Server
3) Database
4) Machine Learning (ML) System
5) System Master

The system will:

1) For each topic, crawl relevant websites to find recent articles about the topic.
2) Pass each article through the machine learning system to determine what the article is discussing.
3) Save this information in the database.
4) When an end-user requests to view the classification for some category, load this information from the database to the web server and display it to the user.

Substructures:

1) Web Crawler:

We must take an innovative approach to make this idea a reality. Our application will have different compartiment:

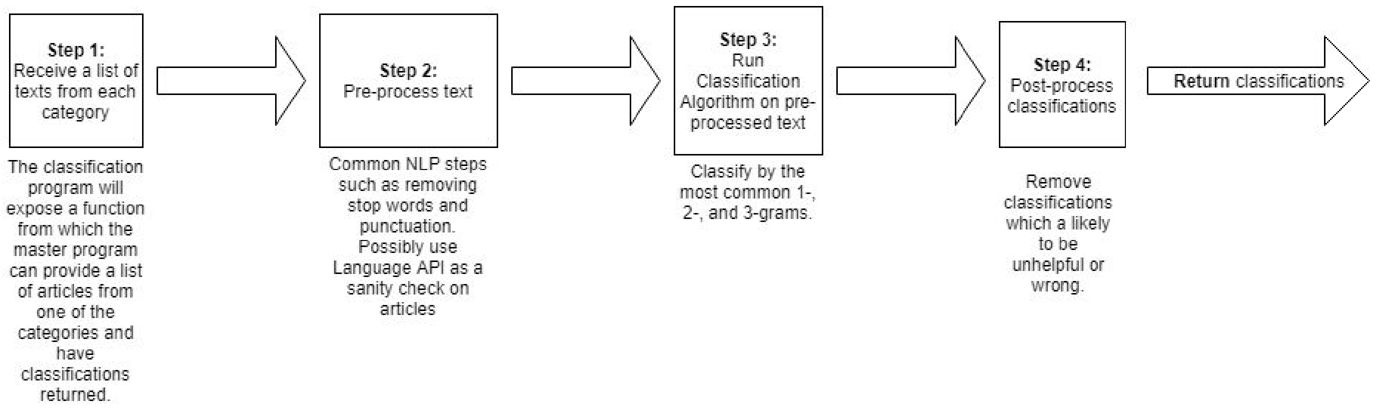
- An efficient crawling system that will crawl any kind of popular web content: Knowing that any popular web platforms have links to other web platform. Our crawling system will recurse throughout these links to get more information.
- A ML system: we will use neural network to make sure that we can filter for the information that we want.
- The information that will be collected will be save in database design to support requests from our system.

What need to be implemented:

1. Crawler
   - (a) Functionalities
     - (i) Feed the data crawled to a database
     - (ii) For the crawling system we are going to use Scrapy library(a python library)
     - (iii) With this Crawling library we can associate a non-relational database (MongoDB) to store our crawling data since the data we are crawling is not structured
     - (iv) We are going to target specific Web content such as Sport and Entertainment Web contents.
     - (v) The crawler library we are going to use got some built-in functions that could be used to avoid duplicate contents. It is the matter of using those function appropriately for more accuracy and efficiency
2. Web Server
   - (a) Since the main goal is to implement a web platform, we are going install an apache web server in a virtual instance in the Google Cloud
   - (b) Aso with this instance we are going to install a web a MongoDB server to integrate that would allow to set up a configuration with a MongoDB Database
   - (c) We are going to install the necessary packages that handle our development of the backend and frontend
3. Database
   - (a) As we have said, we are going to install MongoDB server associated with our web server
   - (b) We are using a MongoDB database because of the unstructureness of the data we will crawl
   - (c) Since the data unstructured we will a store a collection of the data for each information type. For instance there will a collection of data stored in our database in:
     - (i) Sports
     - (ii) Entertainment
   - (d) We will using different MongoDB functionalities such as:
     - (i) MongoDB aggregation Framework
     - (ii) MongoDB BSON format to format our data
     - (iii) MongoDB Capped Collections to maintain our data insertion order in the database
     - (iv) MongoDB Ad hoc queries to get data from the database
   - (e) MongoDB als got some functionalities in terms of scalability, so we have
     - (i) The Sharding function can allow MongoDB to handle
       1) Deployment with large dataset
       2) Intensive throughput actions
4. System Master
   - (a) We will create main program where we will include

(i) The crawler program to get the data and insert it in the database
(ii) The program that will to get the data from the database
(iii) The text classification program to remove duplicate articles, classify the crawled data using Ngrams; possibly augmented by the Google Cloud Language API.
(iv) Create functions to display data from on our platform

(5) Big Data Processing Unit (provided by Google as machine learning API), how to connect with the database
   (a) DataBase
      (i) Our two main concerns are storing data efficiently and accessing data quickly
      (ii) How to store data: NoSQL options seem to have advantages over SQL options in a cloud environment - See scalability; BigTable, Datastore and Firestore (in Beta) are some examples of NoSQL databases that are in the Google Cloud Platform.
   (b) **Text Classification**:
      (i) Classification Process:


Step 1: Receive a list of texts from each category
The classification program will expose a function from which the master program can provide a list of articles from one of the categories and have classifications returned.
Step 2: Pre-process text
Common NLP steps such as removing stop words and punctuation. Possibly use Language API as a sanity check on articles
Step 3: Run Classification Algorithm on pre-processed text
Classify by the most common 1-, 2-, and 3-grams.
Step 4: Post-process classifications
Remove classifications which a likely to be unhelpful or wrong.
Return classifications

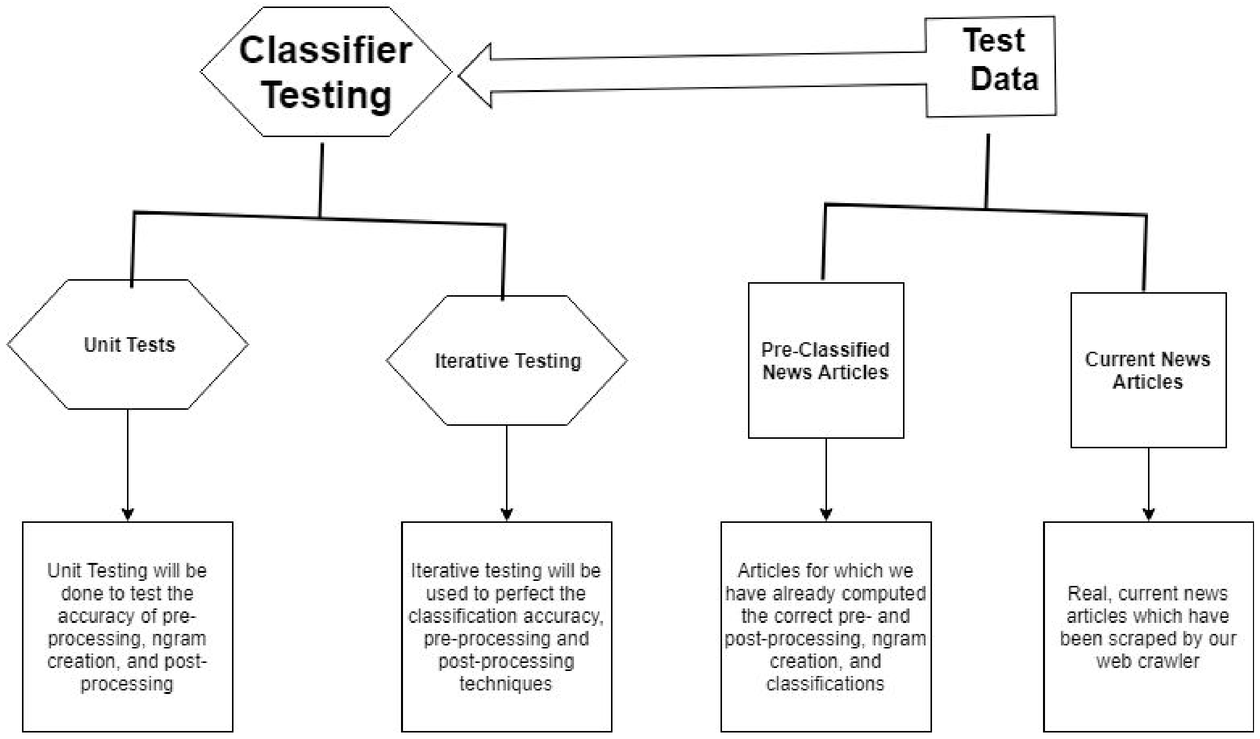
      (ii) **Preprocessing Task: Removing Duplicate Articles**
         1) The presence of duplicate/partially-duplicate articles may negatively affect classifications by over-weighting certain subjects. So, we need a method to find and remove duplicates.
            a) As we will have a large amount of text, this method will need to be very efficient/speedy.
         2) RemoveDuplicates:
            a) Create a generalized suffix tree from the articles
            b) Find all pairs of articles with common substrings of length at least *k*.
            c) For each of these pairs, remove the substring from one of the articles.

3) *k* needs to be large enough not to capture substrings which are like to co-occur by chance or common subject matter, but small enough not to miss genuinely duplicated sections of articles.
    a) *k* = 20?
    b) *k* will be set after testing on different values

(iii) **Classification Will Be Done by 2 Techniques**:

1) NGrams
    a) This has given reasonably accurate classifications for what the texts are discussing. An example would be “Lebron James” in sports.
    b) It is also much more specific than the Google API.
    c) It will require some (basic) fine-tuning though. Non-descriptive but common words are one problem that must be solved (“points” in the Sports category)
    d) After these are dealt with, though, this will provide an accurate classifier for our purposes.
    e) It is also a very fast algorithm.
    f) And, does not have to deal with either the added latency or monetary cost of using the Google API.
    g) As N-grams are both more specific and faster than the language API, they will be used as the main classifier
2) Google Cloud Language API
    a) This is much more accurate than n-grams, but classifications of texts are much too general (“Sports” is about as detailed as it gets)
    b) Still, this may be useful for checking that articles we our texts actually belong to the category we expect it to (either Sports or Entertainment)
    c) The accuracy of the API may be useful, but due to the overly general classifications it provides, cannot be the main tool.

(iv) Testing


Classifier Testing
Test Data
Unit Tests
Iterative Testing
Pre-Classified News Articles
Current News Articles
Unit Testing will be done to test the accuracy of pre-processing, ngram creation, and post-processing
Iterative testing will be used to perfect the classification accuracy, pre-processing and post-processing techniques
Articles for which we have already computed the correct pre- and post-processing, ngram creation, and classifications
Real, current news articles which have been scraped by our web crawler

(6) Main control program
   (a) Define the steps of the control program
      (i) The main program will be interacting like an API to manage the whole system. When a request is made for a specific functionality, the main program will call the appropriate function to execute the following tasks:
         1) Setup a trigger to crawl the data
         2) Setup mechanism to directly store the data crawled in the database
         3) Pull the data for analysis using ML algorithms
   (b) Handling scale out
      (i) Delete old data
      (ii) Load balancing
(7) Display
   (a) How can it be displayed
      (i) We could create generic placeholder that will be updated automatically when the data is refreshed
      (ii) How can it be displayed automatically

II. Scalability requirements

(1) Processing load based scalability
(2) storage/database size-based scalability: NoSQL databases are better at scalability

[according to Vaquero *et al* in “Dynamically Scaling Applications in the Cloud”].

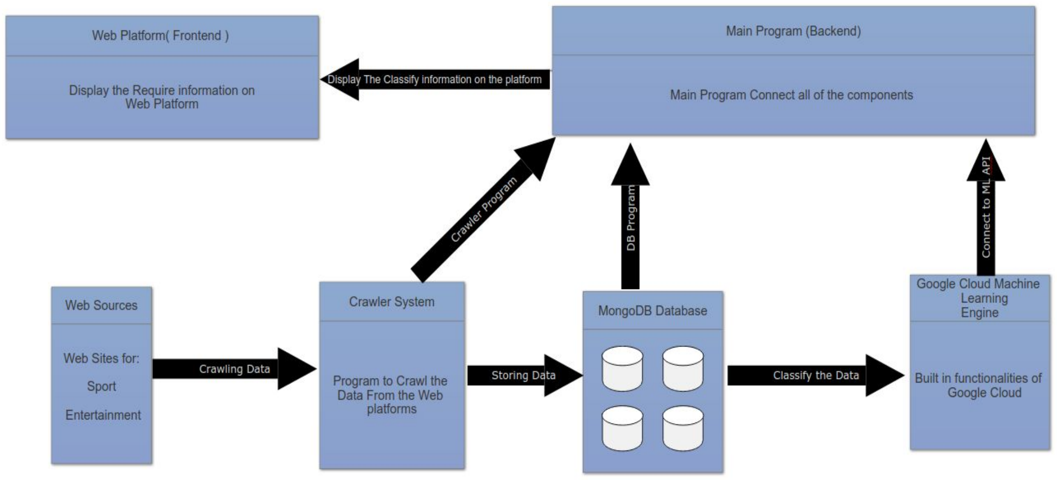
(a) Scale out/in

(b) Factors: Consider costs of storage options (https://cloud.google.com/products/storage/) and if possible, how much data is expected per day on average when determining this.

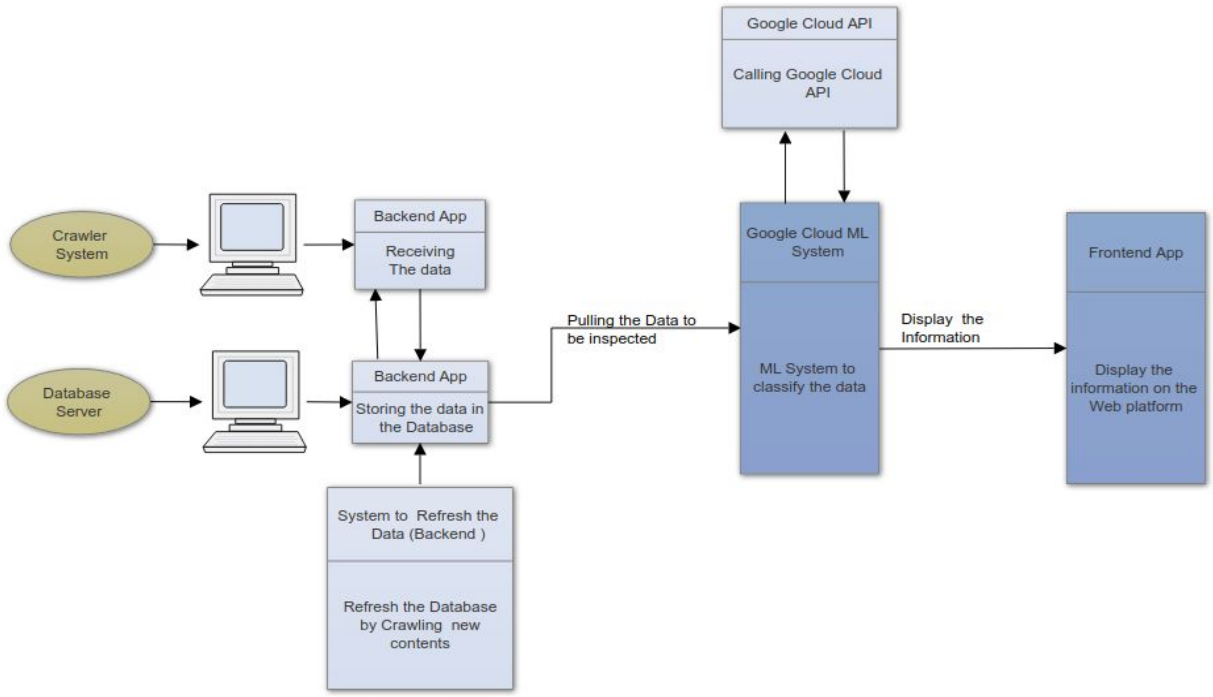
III. System Design

1. The overall system Design

a. First we are going to crawl the data using the crawler function
b. Store the data in a database
c. Set up a function to refresh the database
d. Fetch the data from the database using MongoDB functions
e. Use the Google Cloud Machine Learning system to classify the raw data
f. if the information reaches appropriate level of accuracy display the data on our platform


Web Platform(Frontend)
Display the Require information on Web Platform
Display The Classify information on the platform
Main Program (Backend)
Main Program Connect all of the components
Crawler Program
DB Program
Connect to ML API
Web Sources
Web Sites for:
Sport
Entertainment
Crawling Data
Crawler System
Program to Crawl the Data From the Web platforms
Storing Data
MongoDB Database
Classify the Data
Google Cloud Machine Learning Engine
Built in functionalities of Google Cloud

i. Data workFlow Diagram


Crawler System
Database Server
Backend App
Receiving The data
Backend App
Storing the data in the Database
System to Refresh the Data (Backend)
Refresh the Database by Crawling new contents
Pulling the Data to be inspected
Google Cloud API
Calling Google Cloud API
Google Cloud ML System
ML System to classify the data
Display the Information
Frontend App
Display the information on the Web platform

Write out the descriptions about the whole procedure related to the graph in detail.

Questions about the design:

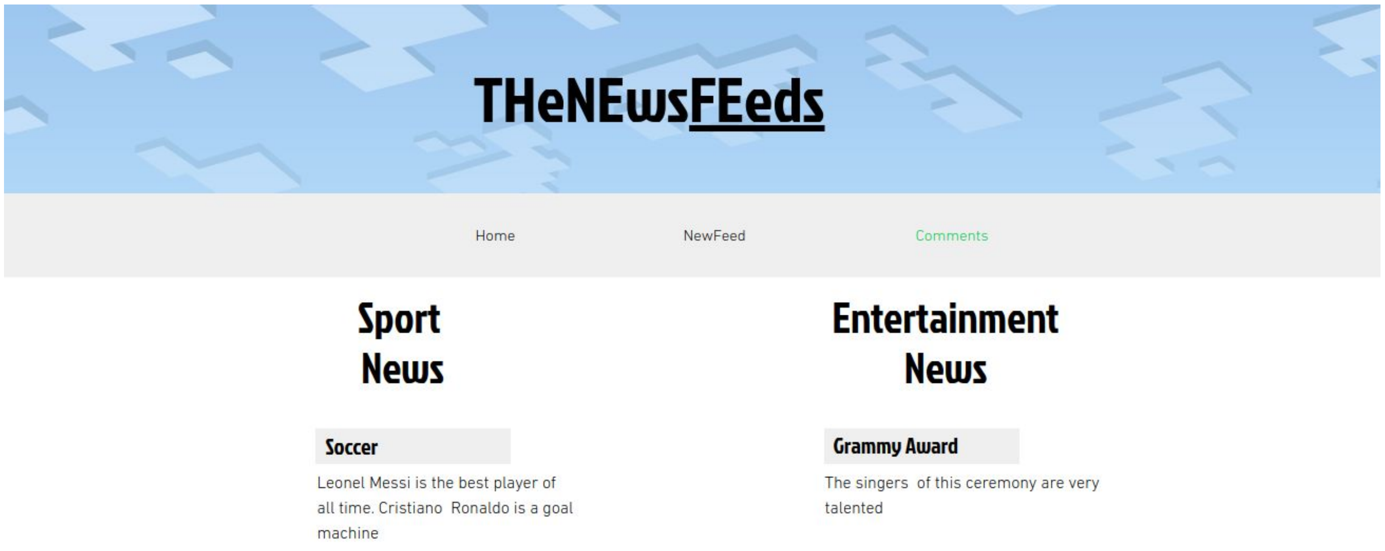
1. **How does the crawler script ensure the efficiency such as avoid duplicated files to be saved?**
2. **Implement storage based scalability or processing load based scalability.**

3. **Host of main program, crawler program, MongoDB database, Machine Learning component… Should they be together or could be separate.**

MongoDB and the crawler system should be one type of VM instance template, so it would be easier to crawl and store data in more than one region. There will one of these per region chosen. Also, having more than one VM with MongoDB avoids having a single point of failure. The crawler will use the diff utility to check for duplicate articles being crawled by a given VM instance before storing in MongoDB. This may require storage buckets for temporary storage. Articles will be deleted after two days.

Machine Learning should be another VM instance template, so processing can scale. The Machine Learning VMs will likely need a larger amount of memory compared to the other VMs, since memory is generally the bottleneck to CPU operations. Therefore, the load balancing scheme will rely on memory usage. The load balancer will be using an internal IP address to ensure connections to the web server component(s) (https://cloud.google.com/load-balancing/docs/internal/), which also simplifies firewall rules with respect to the other VMs. The web server should be another VM template, so it can scale with front-end demand. The load balancer to the web server(s) will prioritize traffic redirection by region.

ii. Web platform design


THeNEwsFEeds

Home NewFeed Comments

Sport News

Soccer

Leonel Messi is the best player of all time. Cristiano Ronaldo is a goal machine

Entertainment News

Grammy Award

The singers of this ceremony are very talented

1. The above illustration is showing the prototype of the web platform
    a. As we can see we have two majors categories on the platform

        i. We have sports events that will have some news about sports in some specific domains
        ii. Also have entertainment events that would list specific information about this area.
    b. When a user visits the platform the information will be already listed according their newness. The user can use the **NewFeed** button to refresh the web page. Later on we could give the possibility to the user to have an account on the platform.
    c. Also the information would be sorted according to their newness to allow the user to have new information constantly.

2. The feeds will be placed in their respective columns - Entertainment and Sports - tentatively as shown in the below front-end prototype:
    a. Each textbox will have its own “id” attribute set to facilitate loading information into the respective category.