ClaimPortal - Building a Social Media Analytics System for Assisting Fact-Checking

By

Sarthak Majithia

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington
in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE
THE UNIVERSITY OF TEXAS AT ARLINGTON

Supervisor: Dr. Chengkai Li

May 2019
Abstract

We are in a digital era where claims made by people can attract attention and spread like wildfire. Misinformation and disinformation about important social and political issues can be intentional and motive can be malicious. Thus, we built a Twitter monitoring platform, namely, ClaimPortal. It assists its users by searching, checking, and providing analytics of factual claims made by politicians and influential people on Twitter. ClaimPortal empowers users with a search API which enables filtering conditions such as date range, tweets from/mentioning specific users, keyword based search, hashtags, check-worthiness scores, and types of claims. We explain the architecture of ClaimPortal and its back-end data collection and computation layer.
# Table of Contents

1. Introduction  
   - 1

2. System Architecture and Components  
   - 2
      2.1. System Architecture  
      - 2
      2.2. Tweet Monitoring and Storing  
         - 2
            2.2.1. Tweet Collection  
            - 2
            2.2.2. Max_id and Since_id parameters  
            - 3
            2.2.3. Tweet Storing  
            - 4
      - 2.3. Elasticsearch Engine  
      - 2.4. Claim Spotter  
      - 2.5. Detecting Claim Types  
      - 2.6. Claim Matcher  
      - 2.7. Claim Portal API  
   - 3. User Interface Features  
      - 8

4. Conclusion  
   - 8

Acknowledgment

References
ClaimPortal: Building a Social Media Analytics System for Assisting Fact-Checking

Sarthak Majithia
Department of Computer Science and Engineering
The University of Texas at Arlington

Supervisor: Dr. Chengkai Li
Committee Members: Dr. Leonidas Fegaras and Dr. Shirin Nilizadeh
May 2019

Abstract

We are in a digital era where claims made by people can attract attention and spread like wildfire. Misinformation and disinformation about important social and political issues can be intentional and motive can be malicious. Thus, we built a Twitter monitoring platform, namely, ClaimPortal. It assists its users by searching, checking, and providing analytics of factual claims made by politicians and influential people on Twitter. ClaimPortal empowers users with a search API which enables filtering conditions such as date range, tweets from/mentioning specific users, keyword based search, hashtags, checkworthiness scores, and types of claims. We explain the architecture of ClaimPortal and its back-end data collection and computation layer.

1 Introduction

The utilization of certainties and describing stories as an instrument for political mobilization has been a traditional approach to induce people of a general accord. The world is combating with an unprecedented measure of deception which can maneuver our society through the reciprocity of facts, perspective, and power. Suppositions can be educated by facts, or by the intentional disavowal of them if a gathering accord forces one so. Social media platforms, in the meantime, present a new and powerful tool to control assent (Allan Leonard and Alan Meban and Orna Young, 2018).

The impact of social media has widened the wellsprings of information and in this manner the development of our stories. Social media has likewise eliminated an essential channel of a writer as the foundation of truth tellers. A few government officials and assessment producers have seized upon this open door as an activity of intensity.

Misinformation and disinformation about important social and political issues can be intentional and motive can be malicious. Events such as Pizzagate have provided substantial evidence that fact-checking is an essential practice and critical before a false claim results in unfortunate events. In recent years, we have seen a surge of tweets made by U.S. politicians. Twitter is considered as a breaking news platform more than a social media network. 86% of the users use Twitter for news and three-quarters of which do so daily (Tom Rosenstiels and Jeff Sonderman and Kevin Loker and Maria Ivancin and Nina Kjarval, 2015). This acted as a focal point in our decision to study politically charged claims being made on Twitter and its need for fact-checking.

As of late, fact-checking has turned out to be progressively predominant in journalism. This is reflected in the expanding number of fact-checking organizations coming up worldwide. While frequently considered as a journalistic interest adjusted to built up news sources, it has likewise been the focal point of work by NGOs, foundations, and non-media aligned associations. Ways to deal with fact-checking rehearses fluctuate appropriately, with numerous associations sticking to a commitment to “actualities” and their dissemination while varying in how the procedures of fact-checking are attempted regarding claim selection and how the fact-checks themselves are imparted.

A job of a journalist is to give day by day information: to give the exact declaration of the political domain. Be that as it may, the truth is more than this information; it incorporates the accessibility of information from an assortment of sources. Individuals mesh these accounts into extensive importance for themselves.

https://en.wikipedia.org/wiki/Pizzagate_conspiracy_theory
In this paper, we present ClaimPortal, a web-based social media analytics system for assisting fact-checking. ClaimPortal continuously monitors tweets and enables its users to sift factual claims embedded in tweets. The system deploys a number of decoupled processes that involve finding check-worthiness score also known as ClaimBuster score (Hassan et al., 2017b), finding similar fact-checked claims for each tweet by employing fact-matching endpoint of the ClaimBuster API, and detect the topic of the claim in the tweet also known as Claim Type. It provides an intuitive and convenient search interface alongside multiple filtering conditions on date range, Twitter accounts, hashtags, check-worthiness scores, and types of claims.

2 System Architecture and Components

2.1 System Architecture

ClaimPortal is composed of a front-end web-based GUI, a MySQL database, an Elasticsearch engine, ClaimPortal API, and several decoupled batch data processing components (Figure 1). The system operates on two layers: the front-end presentation layer, and the back-end data collection and computation layer. The front-end allows users to narrow down search results by applying multiple filters. Keyword search on tweets is powered by Elasticsearch which is coupled with querying the database to enable additional filters. Moreover, it provides numerous visualized graphs. ClaimPortal’s back-end computation layer focuses on data monitoring, gathering, and processing. This includes processes that involve finding ClaimBuster score, finding similar fact-checked claims, detect Claim Type, bulk insertion of tweets in Elasticsearch engine, set user popularity, set hashtag popularity. Each is explained in detail in later sections. ClaimPortal stays up-to-date with current tweets by periodically calling the Twitter REST API.

2.2 Tweet Monitoring and Storing

2.2.1 Tweet collection

ClaimPortal at this moment focuses on politically-charged tweets but will be expanded to eventually cover all types of tweets. We curated a list of
prominent Twitter handles in U.S. politics that include but are not limited to house representatives and senators in the Congress, governors, city mayors, U.S. Cabinet members, other government officials, and political teams of news media. From the Twitter accounts of CNN, Politico, CSPAN, and more, we have access to publicly available Twitter lists \(^6\) which is like a bookmark containing various Twitter handles. Twitter list API \(^7\) allowed us to easily gather member’s user_id and Twitter handles from these lists. We then made use of the user_timeline endpoint of the Twitter REST API to navigate through each user’s timeline and collect their tweets. More specifically, the system navigated through the historical data of a user’s timeline, which is a one-time process. It is significant to note that Twitter API provides access to a limit of 3,200 tweets per user including any retweets. Deleted tweets are accounted as well, though the actual contents are not provided. The system keeps up-to-date with the data by continuously monitoring newly posted tweets. As of April 27, 2019, ClaimPortal monitors around 3,200 Twitter handles and has collected approximately 6.9 million tweets after being deployed in mid-January 2019. We are working on substantially expanding the curated list of Twitter handles to include more Twitter handles of people in politics.

To collect historic tweets from a user’s timeline, the system makes use of the max_id and since_id parameters \(^8\) of the Twitter API. Twitter user timelines can become exceptionally expansive, so there are limits to fetching the tweets from a timeline in a single request. The system must, therefore, iterate through the historical timeline in order to build a more comprehensive list. Due to Twitter’s real-time nature and users constantly composing new tweets every second, standard paging approach will not be an effective solution. Consider an example of a user timeline containing 10 reverse-chronologically sorted Tweets.

Since new tweets are constantly being added to the front of these timelines, there lies a problem in the pagination approach. One technique could be to study the whole timeline in two requests by way of setting a page size of 5 elements and probing the first page, then the second page (Figure 2). If two new tweets are added to the timeline while the two calls were being made, the second request retrieves two Tweets which were captured in the previous call. In reality, if 5 or more Tweets were added between any two calls, later calls would subsequently retrieve all the Tweets returned from the prior request - making an entire API call completely unessential.

2.2.2 Max_id and since_id parameters

Cursoring \(^9\) is a technique for working with streams of data which can be used to resolve the issue described above. Since the top of the tweets timeline updates frequently, reading in an absolute manner will be ineffective. Instead, ClaimPortal reads the timeline relative to the IDs of Tweets it has already processed. This is performed by making use of the max_id parameter of the user_timeline endpoint of the Twitter API.

---


The right way to use max_id is to have the first request to timeline endpoint with specifying a count. While processing the response, and subsequent responses, the system keeps track of the lowest tweet_id received. Tweet_id is a unique number associated with a tweet by Twitter. This ID is passed as the value of the max_id parameter for the subsequent request, which provides a response with Tweets that have IDs lower than or equal to the value of the max_id value. The max_id parameter is an inclusive parameter, and the Tweet with the matching ID is actually returned again. A simple workaround is to subtract 1 from the value and use it as a new value of max_id as shown in Figure 3.

It takes time for the system to process the historic tweets from a user timeline. It then needs to process new tweets which have been added since the last time the timeline was processed, which can be achieved using the since_id parameter. Consider the same example where tweets 1 through 10 were being fetched. Let’s say that Tweets 11 through 18 were recently composed and added to the timeline since the fetching process began. The since_id parameter enables ClaimPortal to fetch tweets after a certain tweet was published. As shown in (Figure 4), a combination of max_id and since_id parameters is the most efficient way to fetch the whole timeline for a particular user. However, at this time, this method can only return up to 3,200 of a user’s most recent tweets including retweets which is an API limitation enforced by Twitter.

2.2.3 Tweet storing

The Twitter REST API provides us with plenty of attributes related to a tweet. However, the system does not require all of it. In fact, a majority of the API’s response is discarded to keep our database size small and yet sufficient enough to provide the necessary information on the portal. As mentioned earlier in Figure-1 ClaimPortal deploys a MySQL database. It is a highly normalized database. Figure-5 shows the underlying structure of the tables.

The two main tables are tweet_data and user_profile. As the name suggest, tweet_data stores attributes of the tweets collected namely: tweet_id, text, time-stamp, ClaimBuster score, and user_id which is referenced from user_profile table. The user_profile table stores information related to any user that composed the tweets collected or mentioned in it. The attributes of the user stored in the user_profile table are user_id, screen_name which is commonly known as Twitter handle, name, reputable which is an internal measure to differentiate between users monitored by the system and users that were mentioned in a tweet, and the popularity which is a measure of how popular the user is. Popularity score is determined by total number of tweets composed by that user which is stored in the tweet_data table. The user_id attribute is referenced in tweet_data and tweet_user_mentions tables. The table tweet_hashtags stores a referenced column tweet_id and the hashtag, as text, that was mentioned in the tweet. Similarly, tweet_claim_type stores the referenced key, tweet_id and claim_type as varchar. The database also stores information about re-tweets and quoted-tweets, and URLs mentioned in the tweets in respective tables as shown in Figure-5.

For every tweet ClaimPortal monitors, the system first checks if the tweet is already available in the database by calling the tweetAvailable function (Figure-6). The tweetAvailable function takes a tweet as parameter and uses the tweet_id attribute to perform a check against the tweet_data table. If it is found, it returns true otherwise, it return false. Any error encountered during this process is logged for debugging purpose. If the tweet is present, the system moves on to the next tweet. Once the system confirms the tweet is not present in the database, the tweet is checked for its type: retweet, quoted tweet or a regular tweet. Each process involves saving the underlying tweet, retweet, and quoted tweet. Any users that were mentioned in the tweet, or the retweet was found to be orig-
The tweet text stored in the database for processing is stripped off of all Emoji\(^{10}\). This allows the system to work with the tweet text for further processing without running into undesired problems. The entire process is repeated periodically for all user timelines that the system monitors to keep the database dynamic and up-to-date with Twitter content.

Relevant indexes were created to make sure the queries that were needed to be performed, to render the front-end, run fast. Such indexes were added to `created_at`, ClaimBuster score, and `tweet_id` in `tweet_data` table, and `user_id`, and popularity (in `user_profile` table).

\(^{10}\)\url{https://en.wikipedia.org/wiki/Emoji}

### 2.3 Elasticsearch Engine

The system enables a user to search through tweets by keyword. This is achieved by utilizing the power of Elasticsearch in the back-end layer. It is a popular industry tool capable of full-text search which works with schema-free JSON documents. Figure 7 shows the structure mapping for the ClaimPortal system.

![Elasticsearch mapping](https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html)

Elasticsearch supports incremental indexing. However, adding one tweet at a time can become a time-consuming process. It is the responsibility of our system to work around this issue by using the bulk API\(^ {11}\). ClaimPortal deploys yet another parallel process that picks up the tweets from MySQL database and inserts the tweets in bulk in Elasticsearch engine (Andrew Cholakian, 2013). The highest `database id` from the last insertion is recorded and used as the starting point for selecting the tweets from the database for the subsequent bulk insert.

\(^ {11}\)\url{https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html}
2.4 Claim Spotter

Politicians are generally quite vocal and actively tweet making numerous claims about their work and campaign goals. There’s a need to filter out what claims are worthy enough to be fact-checked. In ClaimPortal, each tweet is given a check-worthiness score which is a representation of whether the tweet has a factual claim of which the truthfulness is important to the public. This score is obtained by probing the ClaimBuster API, a well-known tool in the fact-checking community (Adair et al., 2019), developed by a group of researchers at the University of Texas at Arlington. ClaimBuster (Hassan et al., 2017a; Jimenez and Li, 2018) is a classification and ranking model trained on a human-labeled dataset of 8,000 sentences from past U.S. presidential debates. Each sentence was manually labeled into one of the following three categories:

(1) A check-worthy factual sentence contains a factual claim and the general public would be interested in knowing its truthfulness. E.g., “He voted against the first Gulf War;” (2) An unimportant factual sentence has a factual claim that is not check-worthy. E.g., “Two days ago we ate lunch at a restaurant.” (3) A non-factual sentence contains no factual claim. Opinions, declarations, and many questions fall under this category. E.g., “But I think it’s time to talk about future.”

Trained on the aforementioned dataset, the ClaimBuster API provides a check-worthiness score for any given sentence or a group of sentences. The score is a decimal number in the range 0 to 1. The higher the score, the more check-worthy the text is. The API also provides the ability to give a penalty. The penalty is a threshold for a text that is deemed to be of low quality. The default for the endpoint is 0.8. The lower the number, the greater the penalty. ClaimPortal deploys a background task of probing ClaimBuster API for getting scores for text in each collected tweet. This executes in parallel to tweet collection.

2.5 Detecting Claim Types

It is vital for a fact-checking tool to gain insights into factual claims and understand what is being spread, by whom, how often, and whether they are true. To get a complete overview of the claim, we categorize the tweets by the types of claim they make. In the ensuing discussion, we refer to these categories as Claim Types, for simplicity of terminology. With efforts found in (Arslan et al., 2019), a collection of FrameNet frames (Baker et al., 1998) was employed at IDIR Lab. The related work created several new frames specifically for politically charged factual claims. The study of mapping frames to Claim Types (Spiliopoulou et al., 2017) was adopted.

From (Spiliopoulou et al., 2017) eight ACE Claim Types were adopted along with their mapped frames: Business, Conflict, Contact, Justice, Life, Movement, Personnel, and Transaction. In (Arslan et al., 2019) the list was extended by introducing four new event types, namely Comparison, Quantity, Stance, and Speech, and their corresponding frames. More specifically, Comparison is for claims that show entities involved in some sort of comparisons based on some criteria, Quantity presents claims with quantities, Stance is for claims that have entities with viewpoints towards issues, events, etc., and Speech is for claims that communicate some messages in the written or spoken form. ClaimPortal deploys a script to identify these Claim Types for each tweet by mapping identified frames to their corresponding Claim Types. One tweet could invoke multiple frames and mapped to multiple Claim Types.

2.6 Claim Matcher

Claim matching is an important step in the workflow of fact-checking. Given a factual claim, it aims at finding identical or similar claims from a repository of existing fact-checks. The premise is that public figures keep making the same false claims. While politicians may refrain themselves from making outright false claims to avoid being fact-checked, oftentimes they even double down after their false claims are debunked.

ClaimPortal leverages the claim matching function in the ClaimBuster API. The fact-check repository was curated from various fact-checking websites. The system measures the similarity between a claim and a fact-check based on the similarity of their tokens.

2.7 ClaimPortal API

To get access to the various attributes associated with each tweet and underlying data from the Twitter API and ClaimPortal’s data process-
ClaimPortal API was deployed for public access. It is a web service designed in Python using Flask micro-framework. It provides API endpoints for loading tweets onto the web-based GUI, searches for hashtags, and searches for users to apply as from-user and user-mentions filters. User is enabled to filter tweets based on multiple filters, namely: keyword search, ClaimBuster score range, date range, tweets composed by a user, tweets that mention a list of users, tweets that include a list of hashtags, and tweets belonging to a list of Claim Types. The system also provides an option to sort the resulting tweets based on date or ClaimBuster score. Based on the filters and sort option requested by a user, ClaimPortal API queries the database to find the resulting list of tweet ID’s and returns it as a JSON response.

Since a tweet id is a unique number assigned to each tweet by Twitter, the web-portal can use it with Twitter’s card API to dynamically populate the latest activity of the tweets. This enables users to get live status of the tweet, such as the number of retweets, likes, comments, and more. Since we are dealing with millions of tweets, there is a need for paginating the resulting set of tweets. The API provides a way to navigate through the query results by the use of pagination parameter. A sample JSON response of the API for loading the default tweets on the homepage can be seen in Figure-8.

The user_search endpoint of ClaimPortal API takes keyword parameters and uses a simple text search query against the MySQL database entry to provide the most relevant users. User search could be made in 2 ways - either searching for their full name, or the Twitter handle. The system is designed to differentiate between these searches by detecting ‘@’ before the keyword. The order of the resulting set of users is by most popular users first. The popularity measure is calculated by counting the total number of tweets found associated with that user in the tweet_data table. Figure-9 shows a sample response from the API while making a search for a user with keyword “Donald”.

Similarly, the hashtags endpoint of ClaimPortal API can be used to search for tweets that contain a list of desired hashtags. Once again, the system is designed to handle search queries irrespective of the use of '#' before the keyword text.

ClaimPortal API employs a security measure, Cross-origin resource sharing (CORS) policy. It is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served. Additionally, the API ensures additional security by deploying Cross-site request forgery (CSRF) protection that enables protection from the malicious exploit of a website where unauthorized commands are transmitted from a user.

---

14 http://flask.pocoo.org
16 https://en.wikipedia.org/wiki/Cross-origin_resource_sharing
17 https://en.wikipedia.org/wiki/Cross-site_request_forgery
3 User Interface Features

ClaimPortal enables the user to sift through the tweets using multiple filters. The default view on the home page shows tweets ordered by the highest ClaimBuster score, in the past 30 days. For a new user, the web-portal displays a quick tutorial showing each component of the user interface. The various filters are as follows:

(1) **Keyword search:** Allows user to make a text-based search by typing the desired keywords like “climate change” in the search input area at the top. This displays all the tweets pertaining to the search criteria, “climate change”.

(2) **Includes hashtags:** Allows user to narrow the resulting set of tweets by hashtags such as #116thCongress or #Election2020.

(3) **Claim Type:** ClaimPortal enables user to search tweets based on a topic of the claim being made in the tweet (e.g., “CONFLICT” or “STANCE”).

(4) **From these accounts:** With this, users can browse tweets composed by a particular user’s Twitter handle (e.g., @realDonaldTrump).

(5) **Mentioning these accounts:** The search results can be additionally filtered by users mentioned in the tweet (i.e., using ‘@’ to tag a user in a tweet, such as @POTUS).

(6) **ClaimBuster score:** ClaimPortal offers a slider interface to filter results based on a ClaimBuster score range - for example from 0.5 to 0.8. The results are automatically displayed as the slider is updated.

(7) **Date range:** Lastly, the portal offers a date picker to filter tweets based on their creation date.

Figure 10 shows ClaimPortal user interface with search results of a sample query. The sample query contains the following filtering conditions: a claim type ‘Stance’, a range of ClaimBuster score between 0.4 to 1.0, a date range from January 16, 2019 to April 16, 2019, and sorted by latest date first.

4 Conclusion

ClaimPortal aims to be a convenient tool for the general public and fact-checkers for assisting critical, investigative inquiry. It provides a handy search interface using multiple filters to easily sift factual claims made on Twitter and provide valuable fact-checks relevant to the tweet.

5 Acknowledgment

The author would like to acknowledge the guidance from Dr. Chengkai Li and work performed by Fatma Arslan, and Sumeet Lubal at the Innovative Database and Information Systems Research Laboratory, University of Texas at Arlington.
References


Damian Jimenez and Chengkai Li. 2018. An empirical study on identifying sentences with salient factual statements. In IJCNN.
