Linking News to Tweets Data

Ryan Conrad, *MAS DSE*, *UCSD*, Ryan Inghilterra, *MAS DSE*, *UCSD*, Sean Rowan, *MAS DSE*, *UCSD*, Brandon Westerberg, *MAS DSE*, *UCSD*

***Abstract*—With an ever expanding amount of heterogeneous data being generated every year data integration becomes an increasingly challenging and resource intensive task. Much of this data is a result of the rise of social media and the content produced by it. As discussed in previous research, social media discussions can be used as a sensor to determine user sentiment with respect to events covered by traditional news sources. Drawing links between news articles and Twitter posts (tweets) represents a considerable challenge due to the difference in vocabulary and writing style used in each medium. In this paper we propose a method to link news articles to tweets using term frequency-inverse document frequency calculated from a corpus of online news articles.**

I. INTRODUCTION

This paper's layout comprises a team project for the Data Integration class in the Masters of Advanced Studies degree program for Data Science and Engineering at UCSD. The team was given news and Tweet data from February and March of 2018 with the task of linking tweets and news. An original research paper was provided to give reference to modern approaches to tackling the problem discussed in the Abstract section. That research paper is titled 'Bridging Vocabularies to Link Tweets and News' [1]. This particular paper goes through the various stages of the project and its workflow to ultimately generate a data flow pipeline and user interface for clients to interact with the results. The layout consists of data exploration, data loading, preprocessing, Python pipeline creation, pipeline optimization, natural language processing techniques, results, and lessons learned.

II. DATA EXPLORATION

71,114 February news articles were provided along with 64,895 March news articles. The median number of words per article was 258. This can be seen in the histogram provided in Figure 1.

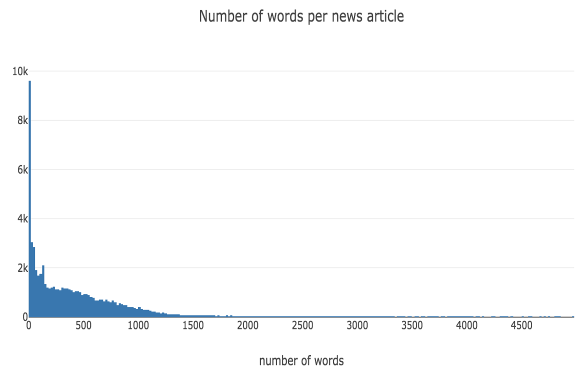


Fig. 1. Words per news article

Tweet data ranged from anywhere from 1 million to 5 million tweets per day. They were provided in smaller files, multiple per day with anywhere from 100,000 to 300,000 tweets per file. Sampling tweet data from a couple different files showed that the number of words per tweet ranged from 1 to around 30 words.

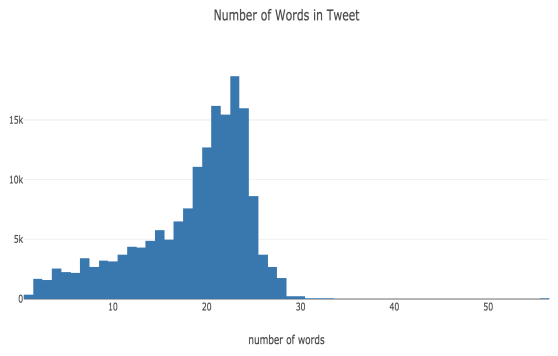


Fig. 2. Average words in a Tweet

III. NEWS DATA

A. Loading News Articles

Raw news data for the project was supplied as a CSV file. An appropriate schema was defined in Postgres for each month's news articles where each months news articles were stored in seperate tables. Using the native Postgres CSV loading utility the raw data was loaded to the database.

A. Filtering News Articles by Character Count

A few heuristics are used to reduce the amount of data being processed and send to downstream steps. The first article selection rule blocks news articles by the amount of characters they contain. Using Postgres a query was written to return the character count for each news article which was visualize with a histogram:

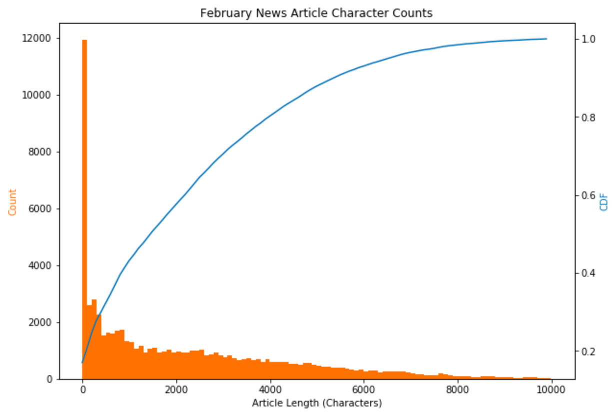


Fig. 3. February news character counts

As seen on the histogram 12,000 news articles (about 20% of the news articles in the month of February) were under 100 characters in length. These articles either contained no text at all or single sentences or sentence fragments. Articles in the 100 - 300 character range were generally very short news updates a couple of sentences long. The blocking strategy rests on the assumption that very short news articles would be more difficult to confidently pair with tweets than a standard high quality article. Looking at the histogram there appears to be a distinct break in articles that are > 500 characters. Upon manual inspection it was found that these were generally full high quality news articles. By setting an article character length cutoff threshold to 500 characters the number of articles sent to downstream processing steps was reduced to ~ 70% percent of its original size. This step was able to be incorporated into a SQL query and pushed down to the database.

B. Filtering News Articles by Character Type

The second blocking strategy used to reduced the amount of articles being processed was to remove articles that contained non ASCII characters. The ultimate goal of this step was to eliminate articles not written in English. By including a where clause in the SQL query that ensures that the text characters are ASCII this eliminates some languages that are no not English. This method is not perfect because some languages use latin text that can be expressed with ASCII, but are not English, e.g. Italian. This method does however eliminate articles comprised of text that is not expressible with ASCII such as Mandarin or Arabic. By utilizing this technique the amount of data can be reduced by another 25%. Using these two techniques news filtering was able to be pushed down to the database level.

IV. PREPROCESSING OF SELECTED NEWS ARTICLES

Once news articles were extracted from the database and pulled into memory preprocessing was conducted to extract the most informative content from each article.

A. Tokenize and Conduct Text Preprocessing

Text processing was conducted using Python, specifically the SpaCy library. SpaCy was chosen for its part of speech support as well as its exceptional processing speed when working with large text corpora. From raw text words were tokenized and reduced to their lower case. Words were then tagged with their corresponding part of speech. Nouns, proper nouns, adjectives and verbs were maintained and all other parts of speech were removed as they were generally non-specific and did not provide any useful context for linking the articles to tweets. Stop words were removed using SpaCy's default English stopword dictionary. Finally each word was reduced to its lemma using SpaCy's lemmatization function.

B. Define set of top 100,000 most commonly used words

To reduce the amount of data being transferred to the similarity calculation steps were taken to reduce the vocabulary of the corpus. Using Python text from all news articles from each month was split and broken into tokens and counted. From this collection only words from the set of top 100,000 most frequently occuring words were saved as tokens to be used to the similarity comparison.

The distribution of the vocabulary appeared to adhere to a power law distribution where the top ranked words were used disproportionately more than those that were ranked lower in terms of use frequency.

V. TWEET DATA

A. Loading Tweets to Couchbase

Tens of millions of Tweets (~40 mil.) were supplied in various .csv files corresponding to a single day split across multiple files. Concatenating these files, each day was now represented as a single file. Using the cbimport command in couchbase, multiple daemons could be started to load the database in parallel into one day per 'bucket' or table.

B. Tweet Query Strategy

The architecture of this projects pipeline and feature needs from tweet data allowed simple queries to be used to push data through a preprocessing pipeline. Each day was contained in a 'bucket' and could be queried with 'SELECT id,text from '<bucket_day>;' to return all tweets for a day to then clean and preprocess within Python.

VI. PREPROCESSING OF SELECTED TWEETS

Similarly to the text content in the news articles tweets were processed in batches before being used for downstream processes. Due to the size of the tweet data being roughly three orders of magnitude greater than the news data the preprocessing was simplified with execution speed in mind.

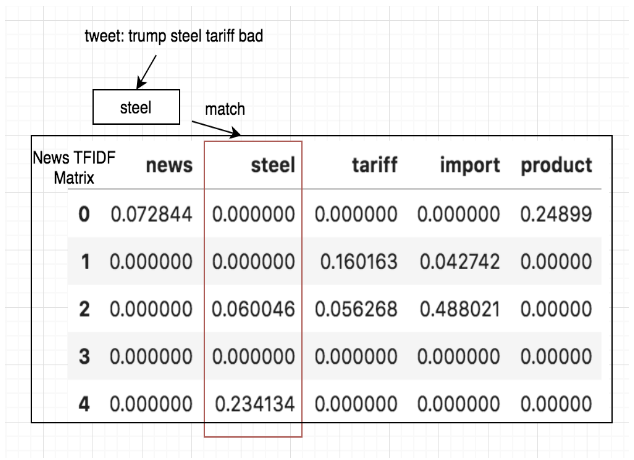
A. Tokenize and Conduct Text Preprocessing

Using Python tweets were pulled into memory and tokenized. Each token was reduced to lower case and a regular expression was implemented to remove special characters. Stop words were removed using SpaCy's default English stopword dictionary.

VII. TWEET AND NEWS SIMILARITY

After trying multiple approaches to calculating tweet vs news article similarity we decided on the following approach:

We first split the news articles by each day and created a TFIDF model for each of those partitions. This provided a matrix where the columns represented the vocabulary of the TFIDF model and each row represented a news article TFIDF vector. We then took each preprocessed tweet and for each word token in the tweet did a lookup on the same-day TFIDF news matrix. The lookup first does a check if the token matches with one of the vocab columns and if it does match, it takes that column and stores it. That column contains all of the TFIDF scores for that specific vocab word for each news article. We keep a running sum of these token column scores. After we do this for every word token in the tweet we will have a similarity score for each news article for that tweet. The intuition of using TFIDF this way is to guarantee that if a news article does come back as a match, it means at least one token from the input string was present in the document. Also, because TFIDF handles relative word importance, each token will have its importance factored in while summing. The performance of this approach is good, since the news TFIDF news article models can be pre-trained for each day, and then iterating through each tweet is just a constant time lookup into the TFIDF matrix. To improve performance and results, we threw out similarity score results that were below a threshold. After manually trying different thresholds, we found a threshold similarity score of 0.3 provided results and performance we were satisfied with.



	news	steel	tariff	import	product
0	0.072844	0.000000	0.000000	0.000000	0.24899
1	0.000000	0.000000	0.160163	0.042742	0.00000
2	0.000000	0.060046	0.056268	0.488021	0.00000
3	0.000000	0.000000	0.000000	0.000000	0.00000
4	0.000000	0.234134	0.000000	0.000000	0.00000

Fig. 4. TFIDF news matrix lookup for a given tweet

We also tried to use pretrained Google News Word2Vec model to transform our tweets and news. The idea was to create a smaller embedding space in order to do the comparisons. Unfortunately, after inspecting the results of the matching, the matches were not intuitive. This was most likely due to the size of the tweet being much smaller than the size of the news article. Using this method also meant that the words in the tweet were not guaranteed to be in news article. There are some benefits from this because it would catch similar word contexts, but the tweets are specific enough to really need an exact word match to conclusively say that a tweet and a news article are related.

VIII. PROJECT PIPELINE

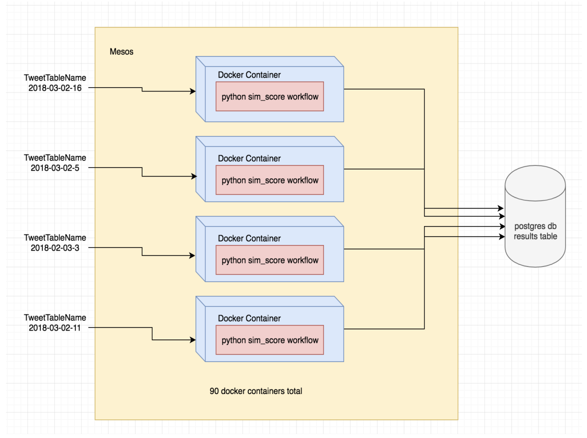
A. Building a Pipeline in Python

Given the large amount of data to process and run our similarity algorithm on, we wanted to consolidate our workflow into a single python workflow file that could easily be configured and run. We combined our code logic into a single workflow.py python file and created a single function that would take in only 3 parameters: news_table, date, and tweet_table. These parameters would simply be the names of the postgres tables containing preprocessed news and tweet data which our workflow would execute on. workflow.py first reads in the tweet and news data out of postgres, filtered only on the date parameter passed in and then executes our TFIDF similarity score algorithm on the data, and finally writes the similarity score results back to a final results postgres table.

B. Workflow Optimization

Consolidating our logic into a single parameterized workflow gave us multiple advantages. Not only did it allow for better code organization and easier runnability, but it allowed us to run our workflow in parallel.

We placed our same python workflow file into 90 different docker containers. As mentioned before, our workflow takes in 3 parameters which are just pointers to the different tweet and news tables in postgres. We ran each workflow file on the docker container in parallel using Mesos, passing in all 90 different news_table, tweet_table, date parameter combinations. Mesos allows for distribution and parallel execution of docker containers across multiple CPU cores. The Mesos cluster we used had over 100 CPU cores available, allowing us to run every one of the 90 docker containers in parallel.


Mesos
TweetTableName 2018-03-02-16
TweetTableName 2018-03-02-5
TweetTableName 2018-02-03-3
TweetTableName 2018-03-02-11
Docker Container
python sim_score workflow
Docker Container
python sim_score workflow
Docker Container
python sim_score workflow
Docker Container
python sim_score workflow
postgres db results table
90 docker containers total
Fig. 5. Processing tweets in parallel with docker and mesos

With this optimization we were able to run our full similarity score workflow on all 90 tweet files in less than 3 hours. Considering that running our similarity score algorithm on just 1 of the 90 tweet tables on a personal computer took over an hour, theoretically running our algorithm on all 90 tweet tables would have taken over 90 hours. The performance boost provided by running in parallel was significant.

IX. INTERACTIVE DEMO

We build an interactive application that allows users to interact with the results of our tweet - news similarity scores. The user interface allows a user to input keywords and select a date. When a user presses the run button, it triggers a python function call. This python function call takes the keyword inputs and date and runs the same TFIDF similarity score algorithm we used for calculating tweet-news score similarities to calculate the news similarity scores of the keyword inputs. The top 5 highest similar news scores are then rendered in an interactive table in the user interface. We used pre-trained TFIDF news vectors which are pulled into memory from python .pkl files during this execution. This allowed for quick calculation and rendering of the results on the UI. Once the news article tables is displayed on the user interface, the user can select a specific news article. This selection will trigger another python function to run. The python function will get the top 5 most similar tweets for the selected news article by running a query against our postgres results table. The results table contains the tweet IDs but no the actual tweets text content. So we also take the tweet IDs of the top 5 similar tweets we queried from the results table and use those ID's to query the preprocessed postgres tweets tables for the actual tweet text content. The results are returned back to the user interface and displayed in an interactive bar chart that displays the tweet and its similarity score. All of this execution completes within seconds, since we have all our postgres table indexed on news IDs and tweet IDs.

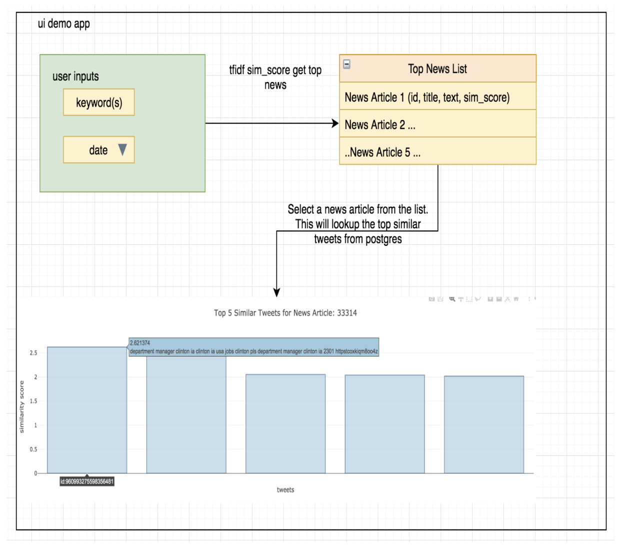

ui demo app
user inputs
keyword(s)
date
tfidf sim_score get top news
Top News List
News Article 1 (id, title, text, sim_score)
News Article 2 ...
..News Article 5 ...
Select a news article from the list. This will lookup the top similar tweets from postgres
Top 5 Similar Tweets for News Article: 33314
2.621374
department manager clinton ia clinton ia usa jobs clinton pls department manager clinton ia 2301 httpstcoxkiqm8oo4z
similarity score
id:960993275598356481
tweets

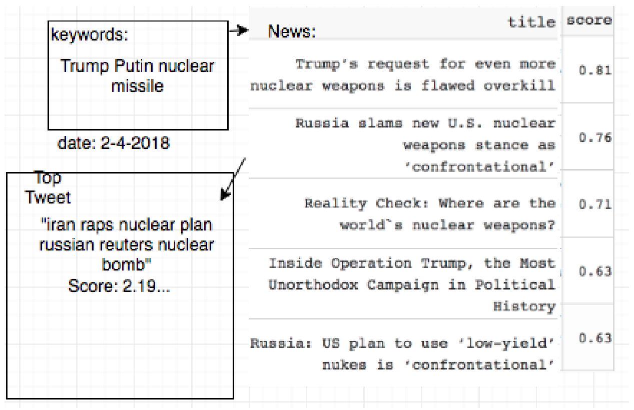
Fig. 6. User Interface Mockup

As explained in detail above, the application we created not only provides a useful and efficient way to interact and visualize our results, but it provides a real-time TFIDF similarity measure calculation of user-inputted keywords for each news article on the selected day.

X. RESULTS

As a way to test if our approach to matching news and tweets were successful we took the top 5 similar tweets for given news articles for a sample of news articles from our postgres results table. We manually inspected the tweets and news articles to see if they how similar they were in content.

Another way we tested our results is by using our interactive demo application. We tried different keywords such as ‘Trump Putin nuclear missile’ on a range of different days and viewed first the top 5 matched news article results. We then selected the different news articles which brought up the top 5 similar tweets by similarity score.


keywords:
Trump Putin nuclear missile
date: 2-4-2018
Top Tweet
"iran raps nuclear plan russian reuters nuclear bomb"
Score: 2.19...
News:

title	score
Trump's request for even more nuclear weapons is flawed overkill	0.81
Russia slams new U.S. nuclear weapons stance as 'confrontational'	0.76
Reality Check: Where are the world`s nuclear weapons?	0.71
Inside Operation Trump, the Most Unorthodox Campaign in Political History	0.63
Russia: US plan to use 'low-yield' nukes is 'confrontational'	0.63

Fig. 7. Example Similarity Results

Manually testing our results, we overall found the matches between news and articles were valid and that our similarity measures worked as expected. There were some cases where we had unexpected match results, most of those coming from articles in different languages matching unexpectedly. Also a note, while our TFIDF approach was able to match on words, it does not match on semantic similarity. For example if a tweet or keyword contained ‘cancer’, it would not match with an article which talks about ‘tumors’ even though they are related.

XI. CONCLUSION

Completing all of the phases discussed in this paper produced an interactive user interface and linkage result stores for news to tweets that the team used to demo to the class. Further result gathering and analysis could be done, yet many trials inputted through the results showed great and intuitive match results. The optimized pipeline and parallelism of processing done for the project proved fruitful, resulting in a complete archive of results to visualize as well as analyze. The next section goes into detail on how the results could be further tweaked, and integration pipeline improved.

XI. LESSONS LEARNED

A. Tweets Data Grooming

The amount of tweet data within the database comprised about 70GB of space, and millions of tweets over the 20 day spans of February and March data. The tweet data could have been cut down on, by filtering the ‘lang’ field to only include “en” for English. Doing so would have most likely increased the accuracy of the project, as well as reducing the size of Tweets to load.

Doing pagination techniques like LIMIT 1000 OFFSET 1000 within Couchbase would also support more dividing and conquering data flow as discussed in the next section.

B. Divide-and-Conquer More

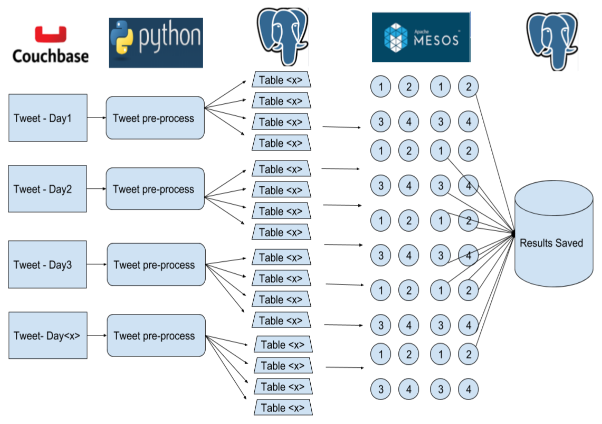


Fig. 8. Pipeline Data Flow of Tweets

In the presentation slide above, the team showed how tweet data was pulled from Couchbase and pushed through preprocessing in parallel, ultimately getting split up into multiple tables per day for a large Mesos cluster[1] to continue the pipeline. Although this proved very fast, it could be improved. The blockage in the flow of data is two-fold. The Couchbase parallelism could have been increased by reading smaller batches of the tweet data to the preprocessing step in Python. This could have been done on its own, or paired with the task removal of Postgres intermediate storage into multiple tables for Mesos to read. The rearranged flow could have divided and conquered the couchbase batches mentioned above, and flowed the preprocessed tweets and loaded them directly on Mesos. Going further, each Mesos docker container could have been hooked up to the Couchbase server, and divide-and-conquer the problem completely within Mesos. Further coding and design time would have made these task re-orderings and removals comprise a better pipeline data flow.

C. Time-boxing News and Tweet Range

The news and tweet data was split over the first ten days of February and first ten days of March. Given the gap between these date timestamps, we could have targeted comparing tweets and news within those months instead of the entire ranges for both. Yet, sometimes stories and tweets are delayed in publishing, so further testing would need to be done to see if results would increase or decrease in quality.

The date ranges of ten days could potentially be a long spread as well for tweet and news linkages. If they were compared within three, four, or five day increments, there might have been different and potentially more temporally relative results in the output of the project.

D. Utilizing NLP within Couchbase

Couchbase uses the concept of Analyzers to provide Full-Text Search and natural language processing capabilities within its data buckets. An expert in Couchbase would have the capability to load news and tweet data within tables in Couchbase and issued these analyzer methods like stemming-porter, apostrophe, to_lower, stop_tokens to preprocess the text data[2]. Even more importantly, other analyzers could be issued in-situ on the tables to apply NLP data science functions between the tables of text, even row by row. This would increase the simplicity of the workflow and portability, yet it is a steep learning curve to have the expertise to apply these in-situ.

E. Golden Data Set

One improvement would be to curate a golden data set of news articles along with the tweets that are referencing those articles. This dataset would be used to measure the performance of different modeling and preprocessing techniques to yield the best result. Without this dataset, it becomes difficult to evaluate the performance between iterations. Human inspection of large datasets is not sufficient.

F. Tweet Word2Vec Corpus

Another improvement could be to train our a Word2vec model using a tweet corpus. The pretrained model we used had been trained using news articles, where the language could be vastly different than tweets. Creating this corpus would take a significant amount of time and extremely difficult to evaluate without a golden data set explained above.

1 http://mesos.apache.org/documentation/latest/architecture/

2 https://docs.couchbase.com/server/6.0/fts/fts-using-analyzers.html

G. Better Similarity Matching With Latent Semantic Analysis

Our similarity TFIDF algorithm does not handle semantic similarity in its current form. Different Latent Semantic Analysis (LSA) techniques could be used in the future to help better capture these type of similarities. Note that while LSA techniques could capture more complex similarities between news and tweets, it is at the cost of performance. With such a large set of tweets, it is not clear that the benefits of using LSA would outweigh the addition of computation time

XII. REFERENCES

[1] T. Hoang-Vu, A. Bessa, L. Barbosa, and J. Freire. Bridging Vocabularies to Link Tweets and News