Social Media and the News Cycle

- **Project Goal:** Build a real time analytics layer for media outlets to assess their influence within social media, including trend detection, brand protection, and awareness of one's own relationship to the social-media universe.

A Core Engine for News Outlet Social Media Investigations

Social Media Sources

Forum threads, **Tweets**, Reddit

Hashtags

#rights
media
#freedomofspeech
#truth
#journalism
#fakenews
#liedetector
#threats
#change
#whitehouse

Core Data Engine

Ingestion | Quality | Analysis | Serving

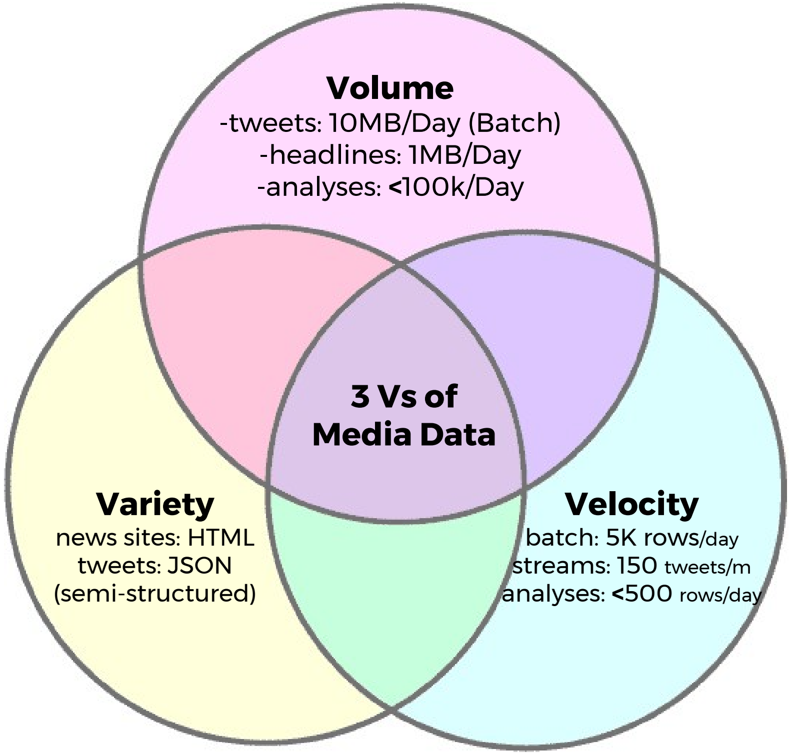
News Sources

Breitbart ,ABC, **PBS**, **NYT**, **FOX**, **CNN**, BBC

Investigative Report

Desired Insights ⟷ Data Characteristics

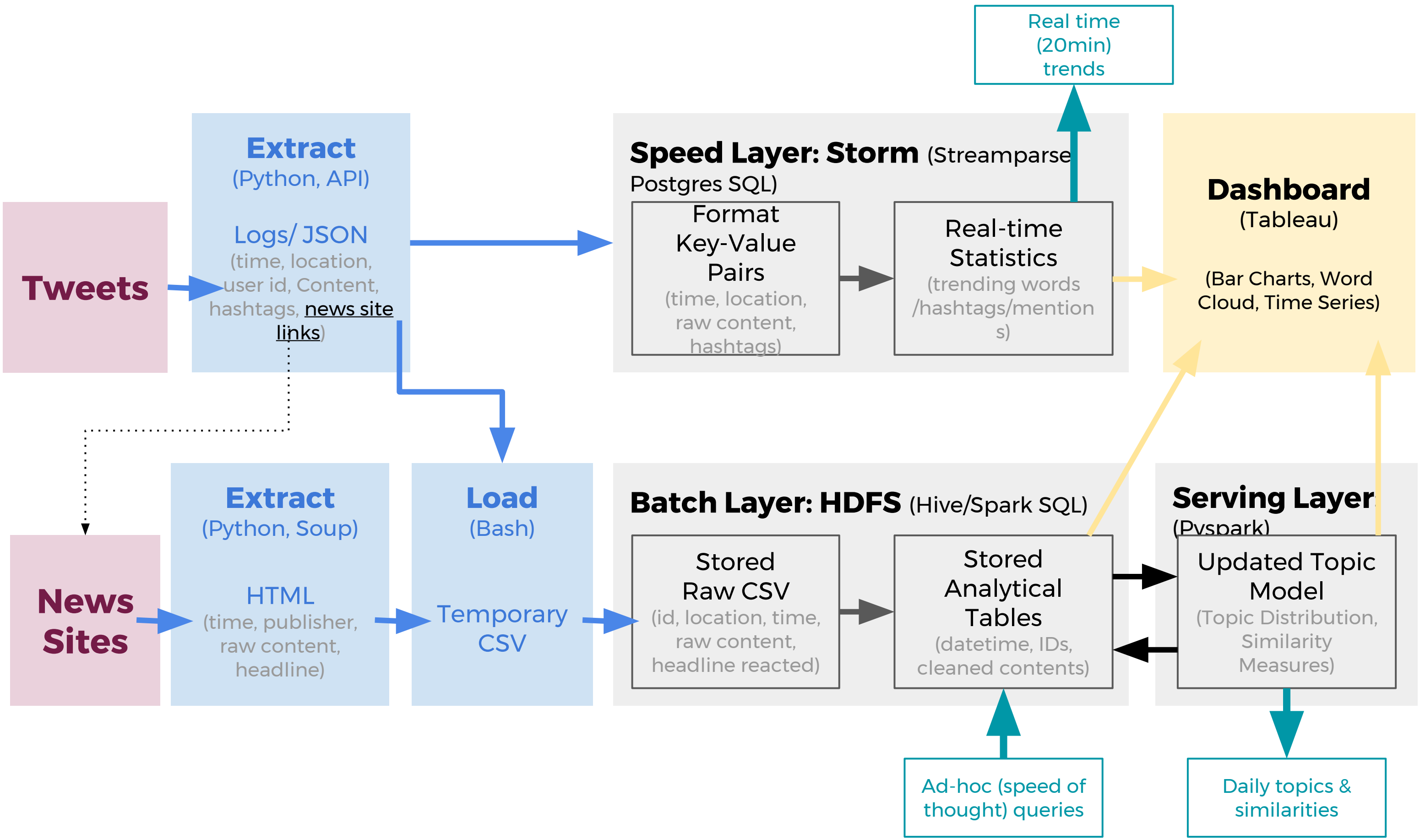
- **Realtime**
  - Trending words
  - Trending @mentions
  - Trending hashtags
- **Every Day**
  - Tweets Topic Summary
  - Headlines Topic Summary
  - Correlation(tweets , headlines)
- **Ad hoc**
  - Popular mentions over time
  - Top influencers by Day/Month/Year
  - Hashtag trends over time
  - Top mentions by top influencers
  - Usual words by top influencers



Data Challenge :

Need an architecture that scales-out for high volume in the long run and ingest high velocity of data

Scale-out solution: Lambda Architecture



Alternatives and Tradeoffs

- **Ingestion and ad hoc queries**
  - Structured Data Store over HDFS
    - More efficient retrieval for queries
    - Less flexibility to manipulate data
    - Less compatible with HTML or JSON (semi-structured)
    - Need Sqoop set up if moving to another HDFS
  - Postgres over Hive or Spark SQL
    - Schema-on-write over schema-on-read: optimized to speed retrieval
    - Efficient Indexing :Fast lookups, lower cost reads
    - Easy to examine query plan
    - Enable insertion and updates
    - ACID database
    - Limited number of records/table
    - Not distributed without sharding
    - Schema modification : expensive to migrate
    - Can cause duplicate materialization
    - Our data is non-transactional and semi-structured
    - Streamed tweet formats are unstable, impossible to impose uniform schema

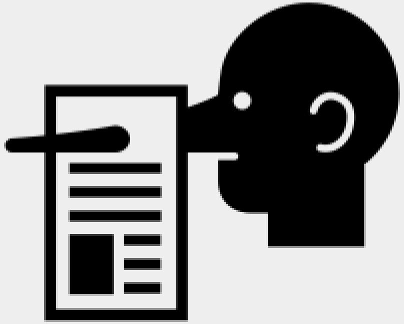
- **Machine Learning**
  - Python over Pyspark
    - Intuitive, easily incorporated with EDA tools
    - Data transformation can be examined more visually pandas
    - Cannot train models with large number of records without RDD advantage
  - R over Pyspark
    - More built-in statistical tools, much easier for model evaluation
    - Single-threaded, only deal with small data problems
    - Distributed R still new in development
  - Scala over Pyspark
    - Better MLlib support in scala
    - New language barrier

- **Streaming**
  - Twitter Heron over streamparse
    - backpressure mechanism, typology scheduling and manage performance by metrics
    - Extra setup time
  - Pyspark D-stream over streamparse
    - Can Run interactively
    - Query over a window of several batches
    - Distributed storage (RDDs)
    - Stateful transformations with list comprehensions
    - Less Stable
  - Postgres over HDFS+Hive for real-time queries
    - Faster queries that favor memory
    - Need migration to HDFS for more persistent storage

Justifications, Tradeoffs

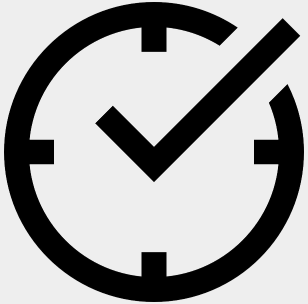
- **Ingestion and ad hoc queries**
  - Scraping and Quality check with Python
    - Compatible with most scraping, and parsing libraries
    - Able to handle our velocity of data
    - Intuitive exception/error management
  - HDFS Storage
    - Easy to scale-out storage with more nodes
    - Compatible with Hive and Pyspark mapreduce, distributed computing jobs
    - Compatible with Spark SQL for speedy queries
    - Resilient nodes and data recovery
    - Higher cost for retrieval
    - Prone to IO bound
  - Spark SQL or Hive for interactive queries
    - Raw data can be subjected to varying interpretation
    - Schema can mutate over-time for new analyses
    - Spark SQL can cache some table for faster reuse
    - Higher costs to read
    - No implicit guarantees about data quality/contents
- **Machine Learning**
  - Pyspark
    - Can push large amount of data through feature extraction and model training pipeline
    - Advantage of RDDs for distributed model training
    - List comprehension similar to python
    - Much harder to debug erroneous RDD processes
- **Streaming**
  - Streamparse
    - Support parallelism by specifying instances in typology code
    - Easy to setup and stable
    - No backpressure mechanism, bad hosts can introduce error when running typology
    - Nimbus master node is a single point of failure
    - Inappropriate use of zoo-keeper for too many writes cause overloading issues
- **Visualization**
  - Tableau
    - Intuitive Interface
    - Enable connection to real-time data through Hive server
    - Take advantage of Data Cubes for fast results display
    - Limited number visualization formats and manipulation of granularities

Further Developments



- **Profile Twitter #fakenews Users**
  - Label tweets and users using ML model real-time
  - Resources/Technology
    - Pyspark 2.2.0 to enable LDA model transformation
    - Convert to Pyspark D stream to outputs streams as RDDs, then feed into Pyspark MLlib

- **Long term Investigative Report**
  - Identify influential twitter users and news agencies for the Trump administration
  - Resources/Technology
    - Migrate data over to S3 storage, only spin-up Hadoop clusters for latest datasets or new analyses over old data
    - Consistently run the system for another 42 months



- **Working Real-time Visualization**
  - Use Hive server for real-time Tableau view
  - Resources/Technology
    - Work out stable EC2 to local machine connection through Hive server

For the following slides, please refer to the video for a full technology demo

Data Ingestion Process

Data Collection

- Use Python scripts
- Scrape News Sites
- Pull Tweets via Twitter API
- Schedule Scripts via Crontab

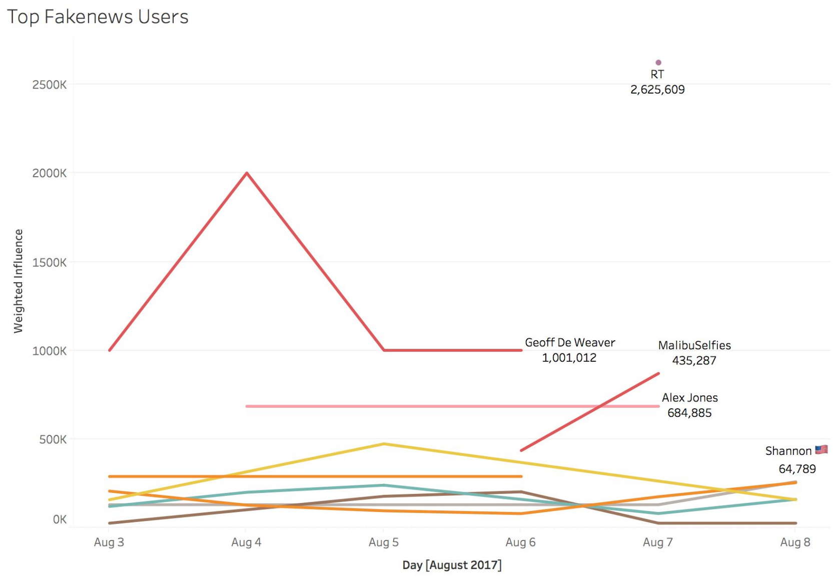
Data Processing

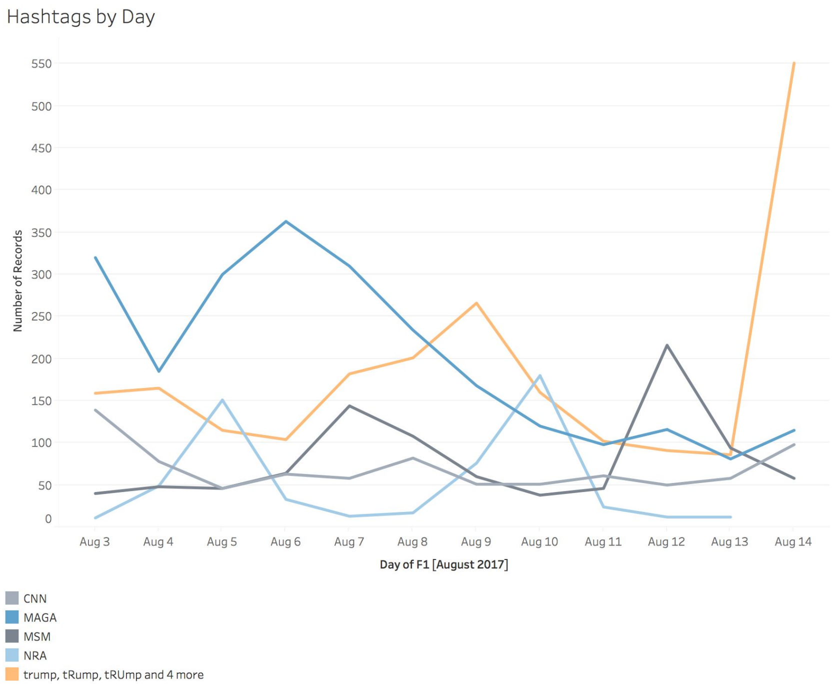
- Python scripts parse, clean incoming data
- Python Scripts format data to be placed in queryable tables in HDFS

Data Storage

- Bash Scripts build tables in HDFS and move cleansed .csv files into HDFS

Insights from Batch Layer

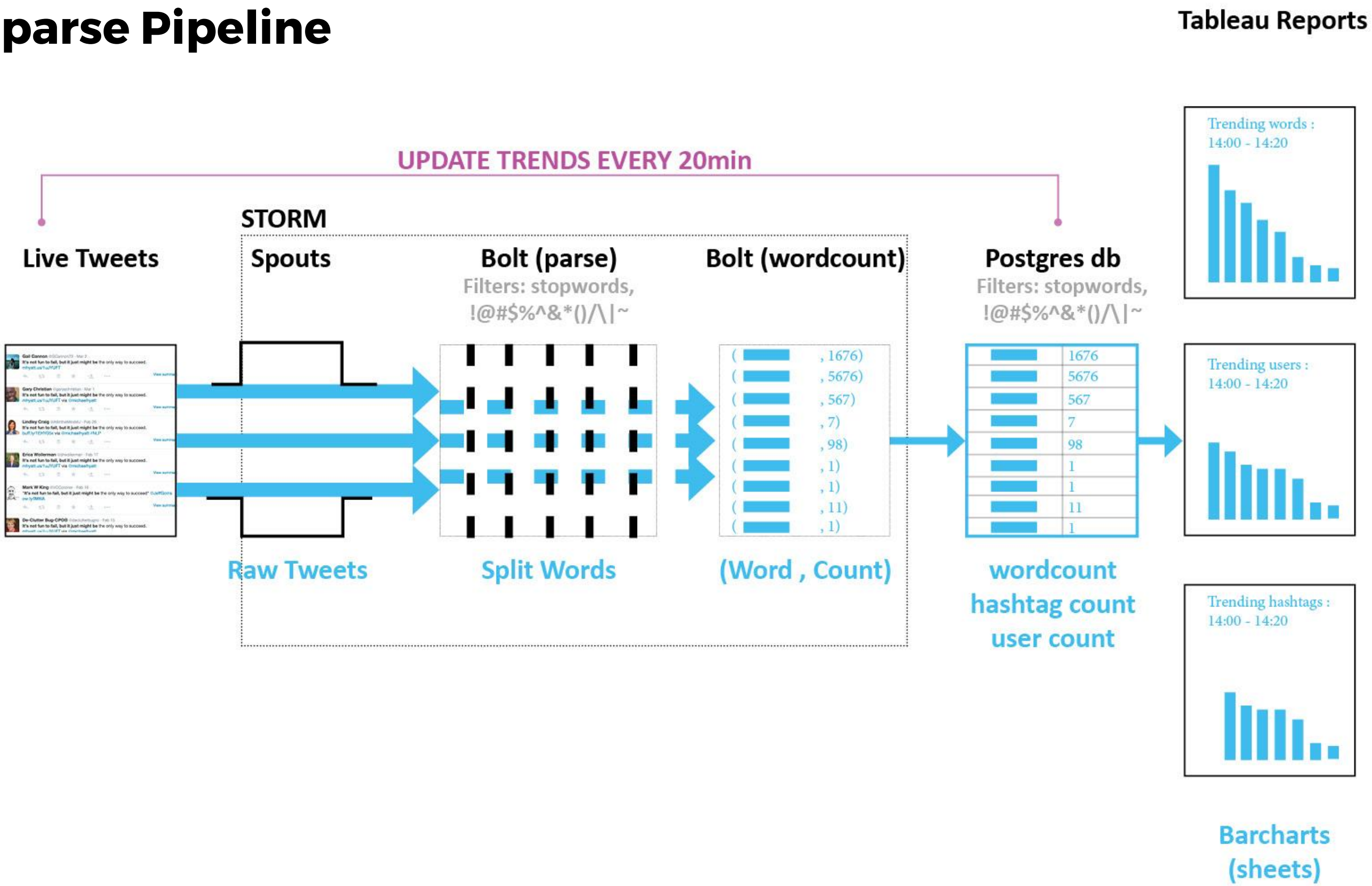




Top Fakenews users by influence (fakenews tweets * followers)

Hashtags closely follow current events

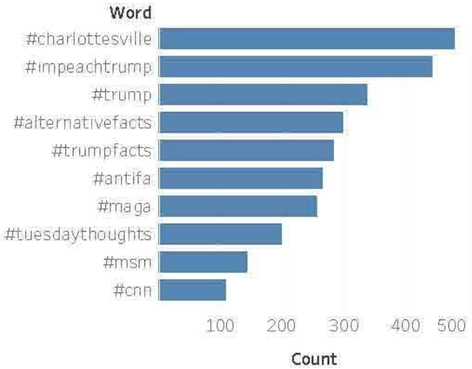
Streamparse Pipeline



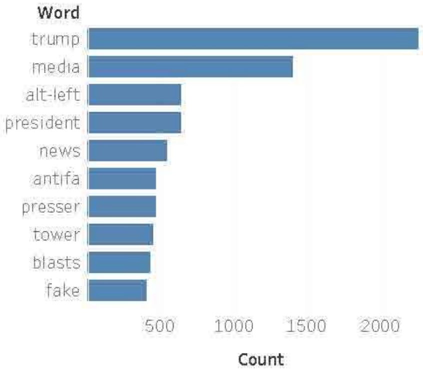
Insights from Streaming Layer

Keep track of trending hashtags, top words and mentions real-time.

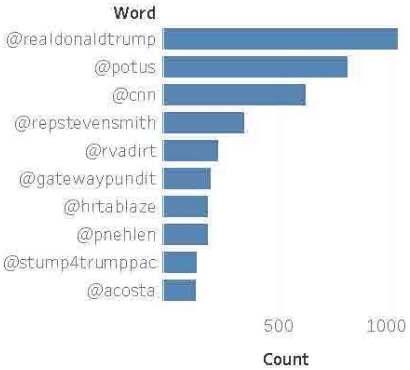
Top Hashtags



Top Words



Top @mentions



Total Retweets

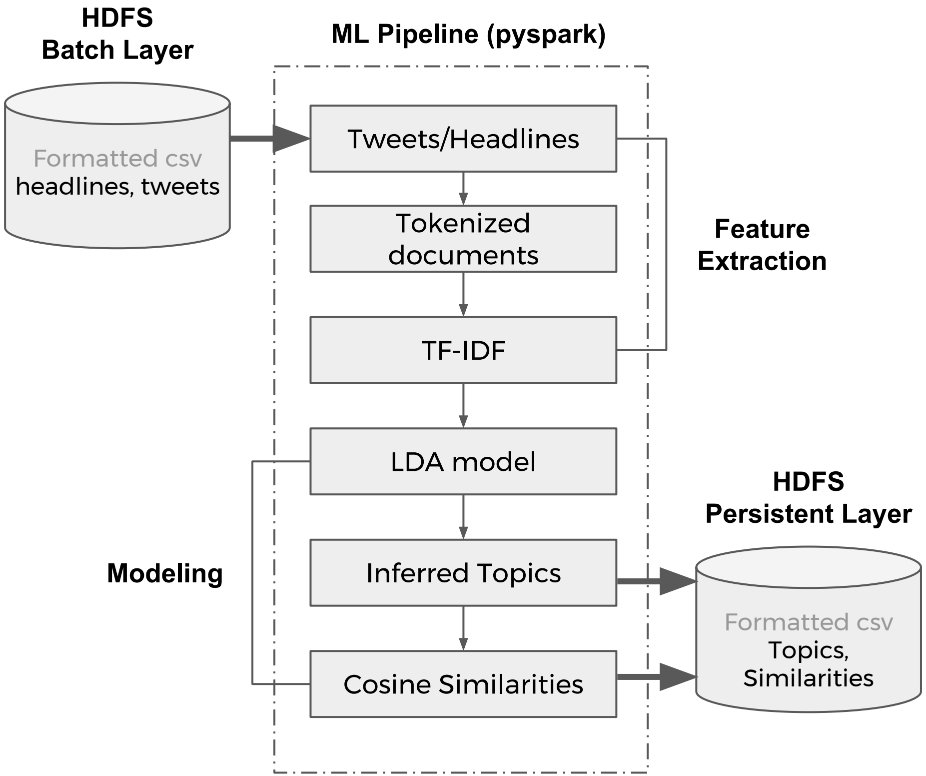
Word	Count
rt	5,480

Total #fakenews

Word	Count
#fakenews	6,590

Data from a 40 minute run on 8/13/2017

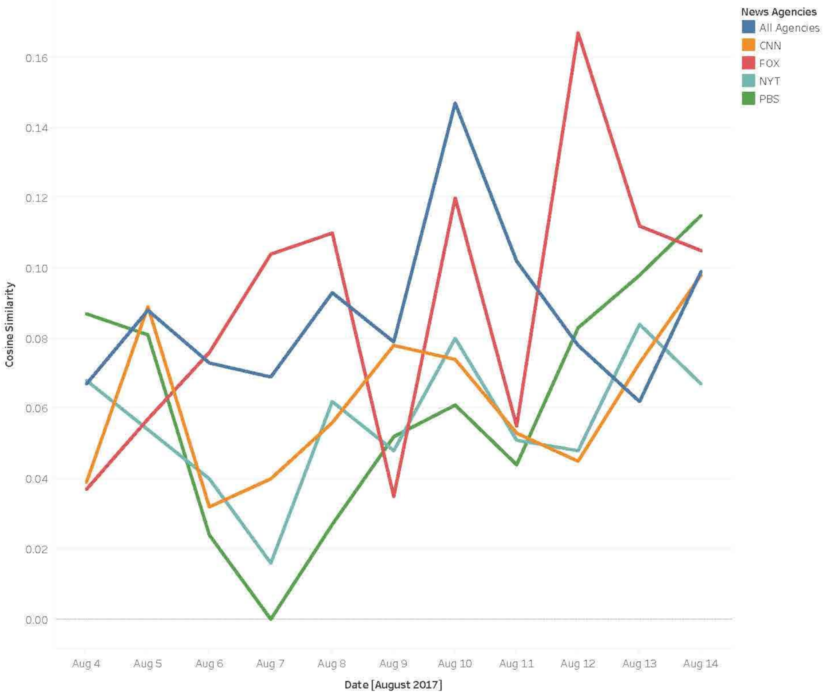
Machine Learning Pipeline



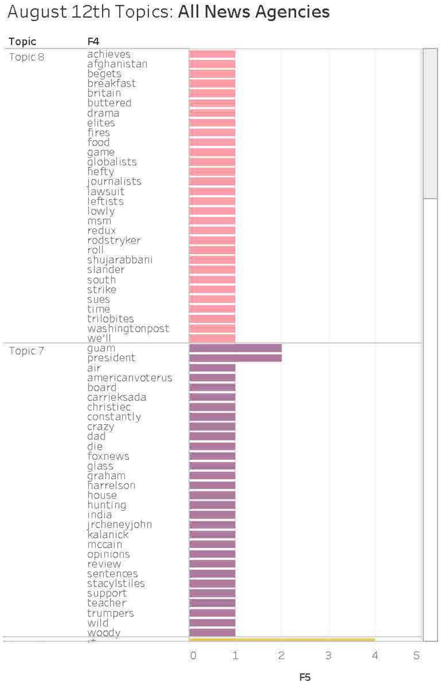

```
def pipeline_headlines(sc, headlines_hdfspath, stopwords_txt_path, n_topics, n_terms, date_range1
= datetime.now() - timedelta(days=1), date_range2 = datetime.now(), news_agency = ["001", "002",
"003", "004"]):

        raw_documents = load_headlines_from_csv(sc, headlines_hdfspath, date_range1, date_
        range2, news_agency)
        stopwords = get_stopwords(stopwords_txt_path)
        tokenized_documents = clean_documents( raw_documents, stopwords)
        tfidf = get_tfidf(tokenized_documents)
        lda_model = train_lda_matrix(tfidf, n_topics)
        topics = get_topics(lda_model, n_terms, tokenized_documents, n_topics

        return topics
-------------------------------------------------------------------------------------------------
def pipeline_cosine_similarity(sc, headlines_hdfspath, tweets_hdfsdir_path, stopwords_txt_path,
n_topics, n_terms, date_range1 = datetime.now() - timedelta(days=1), date_range2 = datetime.
now(), news_agency = ["001", "002", "003", "004"]):

        headlines_topics = pipeline_headlines(sc, headlines_hdfspath, stopwords_txt_path, n_top
        ics, n_terms, date_range1, date_range2, news_agency)
        tweets_topics = pipeline_tweets(sc, tweets_hdfsdir_path, stopwords_txt_path, n_topics,
         n_terms, date_range1, date_range2)

        headlines_words = [word for topic in headlines_topics for word in topic]
        tweets_words = [word for topic in tweets_topics for word in topic]
        unique_words = list(set().union(headlines_words,tweets_words))

        to_counter = lambda words: sorted(Counter([unique_words.index(word) for word in words if
        word in unique_words]).items(), key = lambda pair: pair[0], reverse = False)
        headlines_words = to_counter(headlines_words)
        tweets_words = to_counter(tweets_words)

        counter_to_vec = lambda counter: [pair[1] for pair in counter]
        headlines_vec = counter_to_vec(headlines_words)
        tweets_vec = counter_to_vec2(tweets_words)
        cosine_sim = cosine_similarity(headlines_vec, tweets_vec)

        return(headlines_topics, tweets_topics, cos_sim)
```

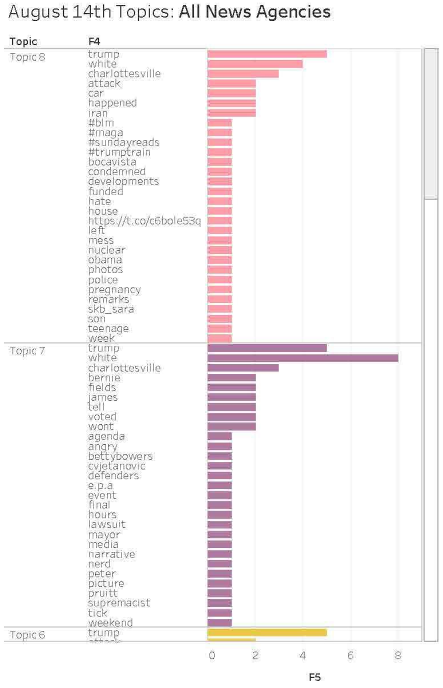
Insights from Machine Learning Layer (1/2)

Use cosine similarities compare how well headlines from each news agency is reacted by tweets.



Examine the inferred topics by the dates to study change of topics.





Insights from Machine Learning Layer (2/2)

Examine the top words echoed by both headlines and tweets. Compare across different news sources.

Tweets vs PBS August 14th

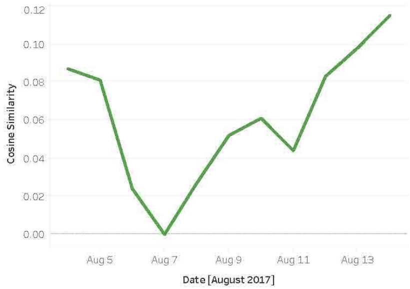
Tweets vs NYT August 14th

Tweets vs FOX August 14th

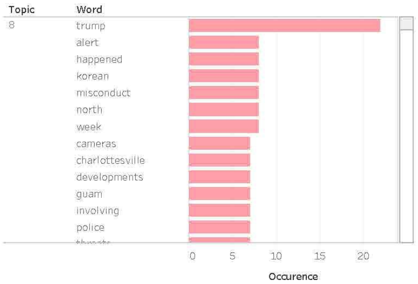
Tweets vs CNN August 14th

Focus on one news source and study its trends and topics further.

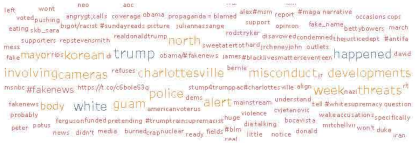
Cosine Similarities PBS vs Tweets



August 14th Topics: **PBS**



Tweets vs PBS August 14th



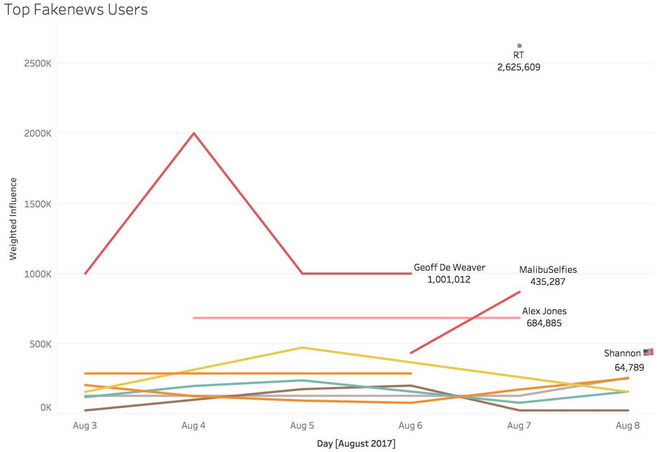
Tweets vs PBS August 13th

Tweets vs PBS August 12th

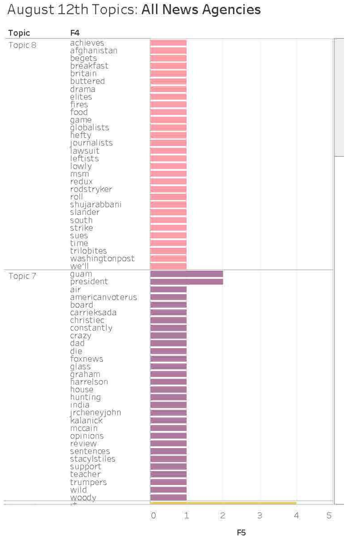
Tweets vs PBS August 11th

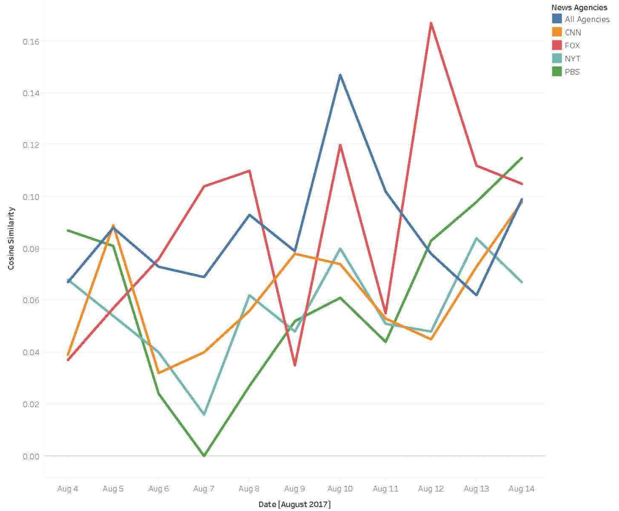
A complete investigative report

Investigate trends and patterns in ad hoc fashion



Study feedback dynamics and correlation between social media and news agencies





Keep up-to-hour with twitter behaviors using summary statistics

Total Retweets

Word	Count
rt	5,480

Total #fakenews

Word	Count
#fakenews	6,590