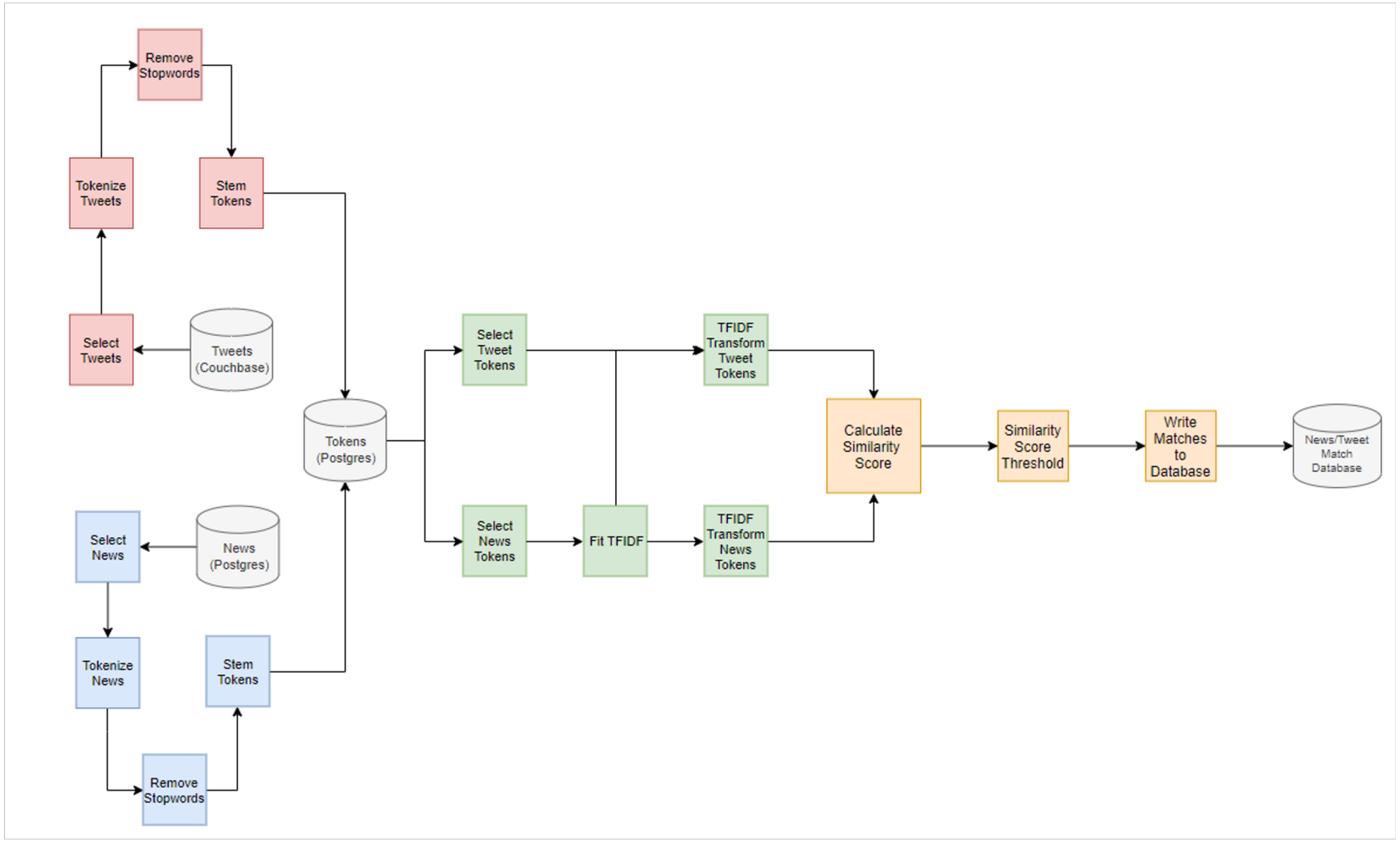
Linking News to Tweets

Ryan Conrad
Ryan Inghilterra
Sean Rowan
Brandon Westerberg

DSE 203 FA18

Intro and Data Flow

- **Goal:** Integrate tweets with news articles based on text content
- **Implementation:**
  - Integrate heterogeneous data sources
  - Apply relevant preprocessing techniques
  - Calculate similarity score for news/tweet pairs
  - Create dashboard to visualize the results for end user



Tweet Storage in Couchbase

- Couchbase uses the concept of Buckets to represent tables
  - Given February and March Tweets, 20 buckets made
  - Each bucket contains a day of tweets
    - approx. 5-8 GB compressed to 2-4 GB
    - Couchbase compresses the JSON in half

- Couchbase as a NoSQL JSON database requires no schema
  - At first, no Indexes are applied, improving load speeds.

- Indexes
  - Primary Index on “id_str” of Tweet data (ID)
  - Secondary Index on “text” of Tweet data (tweet_text)

Loading Tweets into Couchbase

- Using the cbimport:
  - cbimport json -c couchbase://127.0.0.1 -u Administrator -p password -b febday1 -d file:///home/dseboss/share/TweetDataFeb/week1/20180201.txt -f lines -g key::%id_str% -t 4

- cbimport can be used in parallel to load multiple days at a time
  - -b flag is for the bucket to load into
  - -f flag is to load by line
  - -g is to generate an ID, in this case, it uses “id_str”
  - -t is for how many threads for this job

- Loads each day into its corresponding Couchbase Bucket

- Then create indexes:
  - CREATE PRIMARY INDEX ON `febday1`;
  - CREATE INDEX tweet_text ON `febday1`(text);

Tweets Loading and Storage

- As stated, used cbimport in parallel on .txt tweet files merged into tweet days
- Create indexes:
  - CREATE PRIMARY INDEX ON `febday1`;
  - CREATE INDEX tweet_text ON `febday1`(text);
- Total size of database approx. 100 GB

< Impl, ExecEng, Config >

Parameter	Input
Impl	Store Tweets
ExecEng	Couchbase
Config	> 4 GB Memory

Data Flow Factors

Factor	Description
Solution Type	Accurate
Application Domain	Batch
Execution Environment	Centralized

News Loading and Storage

- Raw news data provided as a CSV is loaded in to a Postgres database
- News for each month stored as separate table
- Defined schema:

```
CREATE TABLE News (
news_id primary key,
news text,
src text,
title text NOT NULL,
collectiondate date,
publishdate date,
url text,
author text[],
keywords text[]
) ;
```

< Impl, ExecEng, Config >

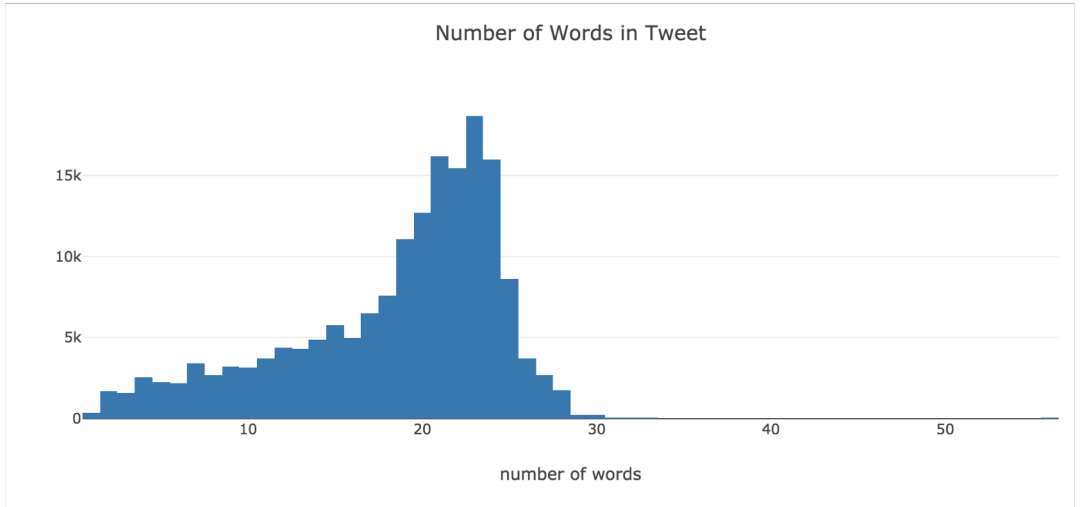
Parameter	Input
Impl	Store News Articles
ExecEng	Postgres
Config	> 4 GB Memory

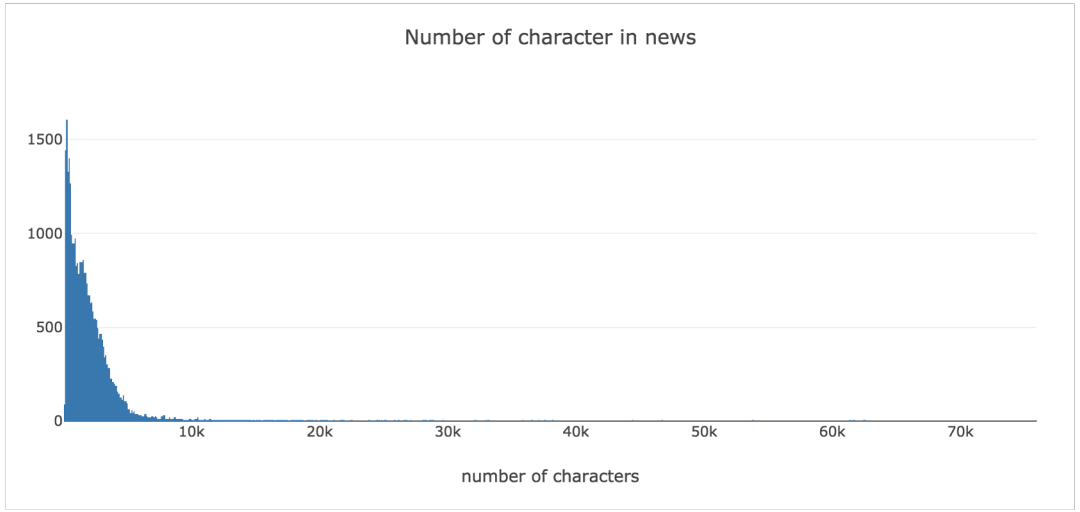
Data Flow Factors

Factor	Description
Solution Type	Accurate
Application Domain	Batch
Execution Environment	Centralized

Data Exploration

- 1 weeks worth of news for February and March each - ~60k articles each month
- tweet data anywhere from 1 million to 5 million tweets per day, broken into small files (multiple files per day)
- sampling tweet data shows that number of words range anywhere from 1-30 words
- news article median length was around ~1500 characters





Selecting Tweets for Preprocessing

- select id,text from febday1;
- Implement this for each bucket (i.e. febday2, febday3, marday1)
- Pushes tweets data into Tweet preprocessing

< Impl, ExecEng, Config >

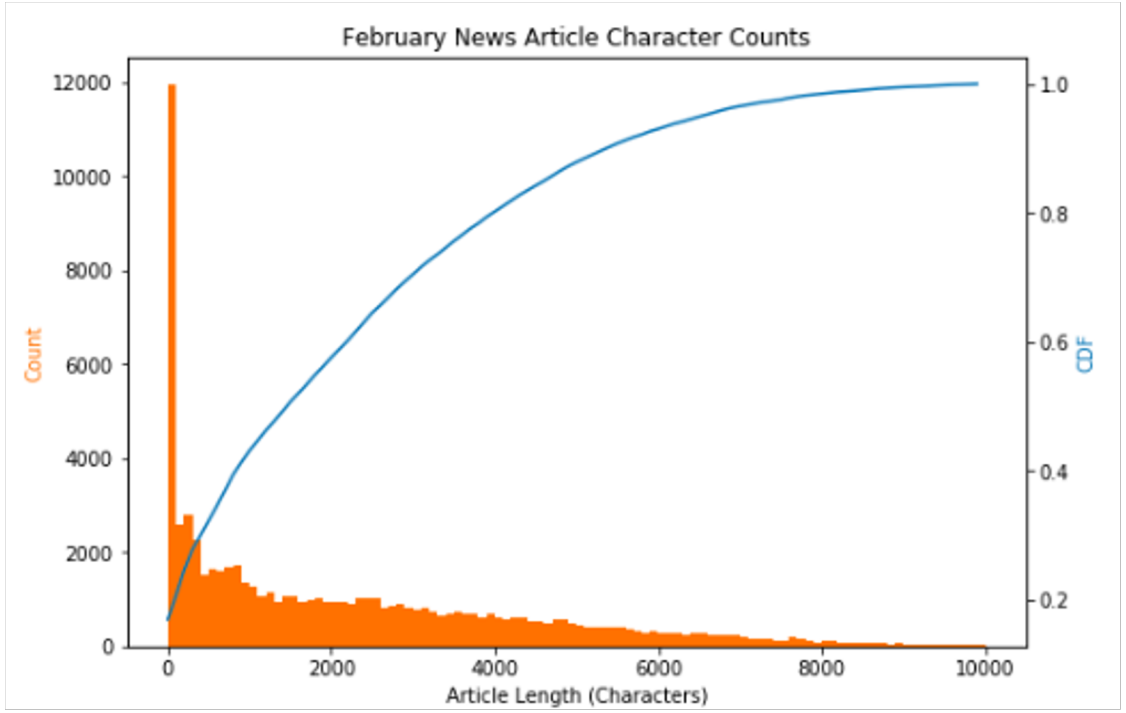
Parameter	Input
Impl	Query Tweets
ExecEng	Couchbase
Config	> 8 GB Memory

Data Flow Factors

Factor	Description
Solution Type	Accurate
Application Domain	Batch
Execution Environment	Centralized

Select News Articles for Preprocessing

- Heuristic 1: Select news articles with > 300 characters
- Heuristic 2: Select news articles with ASCII encodable characters
- Using these two heuristics number of news articles selected was reduced by ~ 25%



< Impl, ExecEng, Config >

Parameter	Input
Impl	Select News Articles
ExecEng	Postgres
Config	> 4 GB Memory

Data Flow Factors

Factor	Description
Solution Type	Heuristic
Application Domain	Batch
Execution Environment	Centralized

Tokenize and Filter News

- Text from selected articles was tokenized and processed
  - Nouns, proper nouns, verbs, and adjectives selected
  - Stop words removed
  - Top 100k words
- Stem of each processed word returned from processing function
- High vertex cost, could be parallelized easily with additional computational resources
- Collected tokens saved to Postgres for intermediate storage

< Impl, ExecEng, Config >

Parameter	Input
Impl	Tokenize/Filter Vocabulary
ExecEng	Python (SpaCy)
Config	> 4 GB Memory

Data Flow Factors

Factor	Description
Solution Type	Approximate
Application Domain	Batch
Execution Environment	Centralized

Tokenize and Filter Tweets

- Text from selected articles was tokenized and processed
  - Regular expressions were used to remove special characters and maintain relevant text content
  - Stopwords removed
  - Words stemmed
- Vertex cost decreased by simplifying workflow for tweet processing
- Collected tokens saved to Postgres for intermediate storage

< Impl, ExecEng, Config >

Parameter	Input
Impl	Tokenize/Filter Vocabulary
ExecEng	Python (NLTK)
Config	> 4 GB Memory

Data Flow Factors

Factor	Description
Solution Type	Approximate
Application Domain	Batch
Execution Environment	Centralized

Calculate Similarity Scores

- TFIDF was used for calculating similarity score
- Fit on news

For each word in tweet:

Sum column at the word index if the word is in the TFIDF vocab

Return top n tweets with the highest sum over threshold t

- Tried using word2vec, but did not perform well

< Impl, ExecEng, Config >

Parameter	Input
Impl	Calculate Similarity Score
ExecEng	Python (NLTK)
Config	Apache Mesos Cluster

Data Flow Factors

Factor	Description
Solution Type	Accurate
Application Domain	Batch
Execution Environment	Distributed

Building an Automated Pipeline in Python

- We wanted to make our similarity score algorithm reusable, clean, and organized.
- Our workflow would only take in 3 inputs: news_table, news_date, and tweet_table. These would all be pointers to our preprocessed data in postgres
- Having this pipeline will allow to easily dockerize our workflow.
- Dockerizing our workflow will allow for optimization by running in parallel (see next slide

running the full pipeline in 1 line of code!

```
In [7]: news_table = 'marnewsprocessed'
        news_date = '2018-3-2'
        tweet_table = 'tweetsprocessed2018_03_02_6'

In [8]: result_df = run_main(news_table, news_date, tweet_table)
```

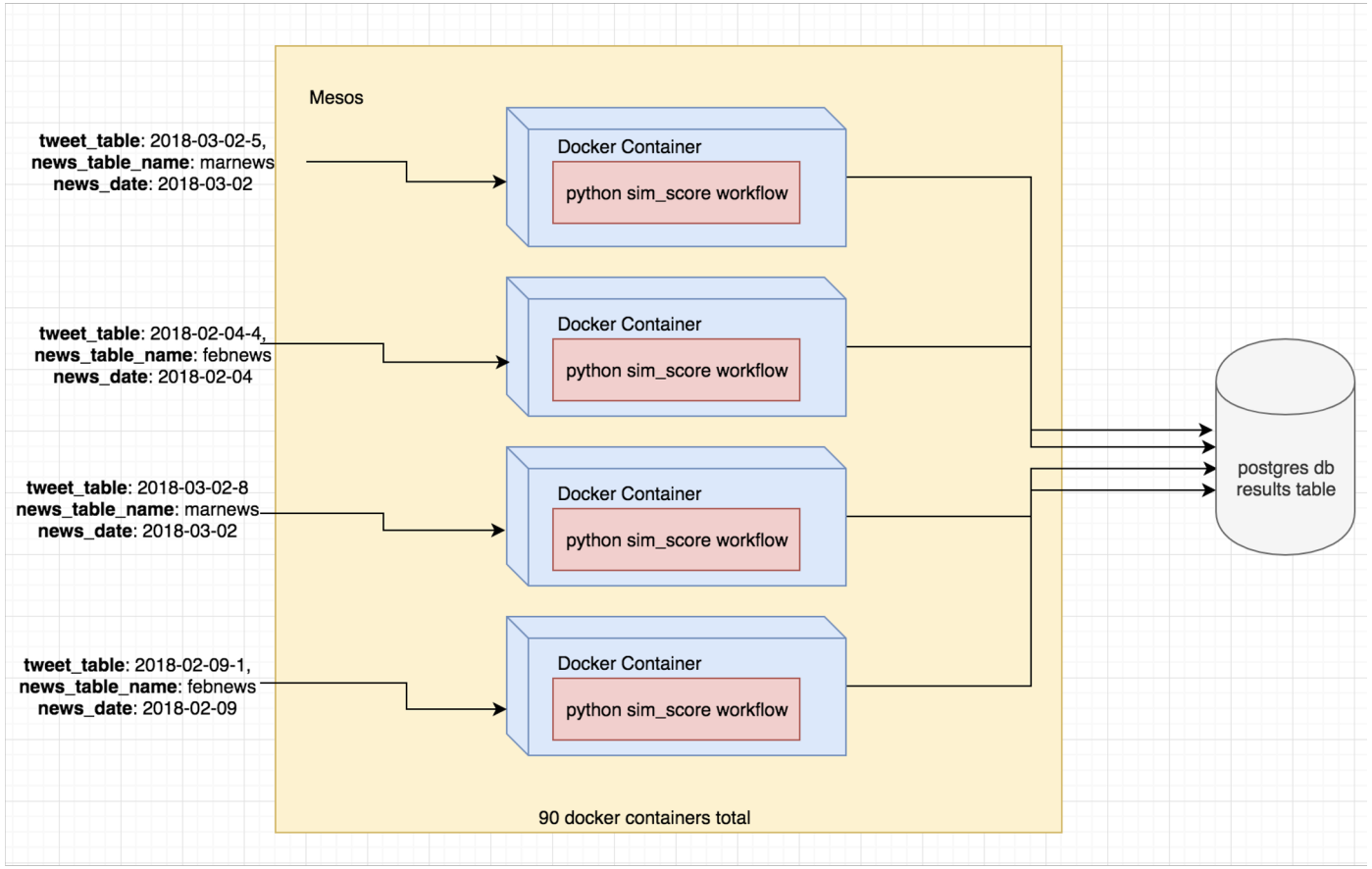
All of the logic organized in functions in a workflow.py file

workflow.py

```python
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk.stem
import sqlalchemy as sq
import io

english_stemmer = nltk.stem.SnowballStemmer('english')
analyzer = TfidfVectorizer().build_analyzer()

PG_CONN_STRING = '******'


def stemmed_words(doc):
    return (english_stemmer.stem(w) for w in analyzer(doc))


def get_tfidf():
    tfidf = TfidfVectorizer(stop_words='english', smooth_idf=True, analyzer=stemmed_words)
    return tfidf


def sim_scores(tweet, news_tfidf_vector, vocab):
    tf_sum = np.zeros((news_tfidf_vector.shape[0], 1))

    # iterate through the words in tweet
    for w in tweet.split():
        #w = w.lower()
        if w in vocab:
            idx = vocab[w] # save idx
            tf_sum += news_tfidf_vector[:, idx]

    output_df = pd.DataFrame(tf_sum)[0]

    return output_df


def full_workflow(tweet, tweet_id, news_vector, tfidf, topN, news_dic, news_df):
    news_scores = sim_scores(tweet, news_vector, tfidf.vocabulary_)

    # want news_id and news_scores associated and then want to max topN values
    news_df['score'] = news_scores

    sorted_news = news_df.sort_values('score', ascending=False)

```

Workflow Optimization – Running in Parallel with Mesos

- Need a way to scale our python similarity score algorithm to run on millions of tweets and thousands of news articles
- Dockerize our python automated pipeline
- Run docker containers in parallel with mesos
- Each container will write to same result postgres table when running complete
- x50 speedup, was able to run our similarity score algorithm on ~50 million tweets and ~150,000 articles in < 3 hours!



Save Similarity Scores to Database

- Each worker in the mesos cluster writes results to the same postgres table
- when all mesos workers were finished we had a postgres table with all of our results.

	newsid	sim_tweets	news_date	tweet_file
0	8025	[(969506880698675203, 0.8943515886747533)]	2018-3-2	2018-03-02-9
1	8025	[(969482547284586496, 0.8943515886747533)]	2018-3-2	2018-03-02-8
2	8025	[]	2018-3-2	2018-03-02-7
3	8025	[]	2018-3-2	2018-03-02-10
4	8025	[(969455634495221760, 0.8943515886747533)]	2018-3-2	2018-03-02-6

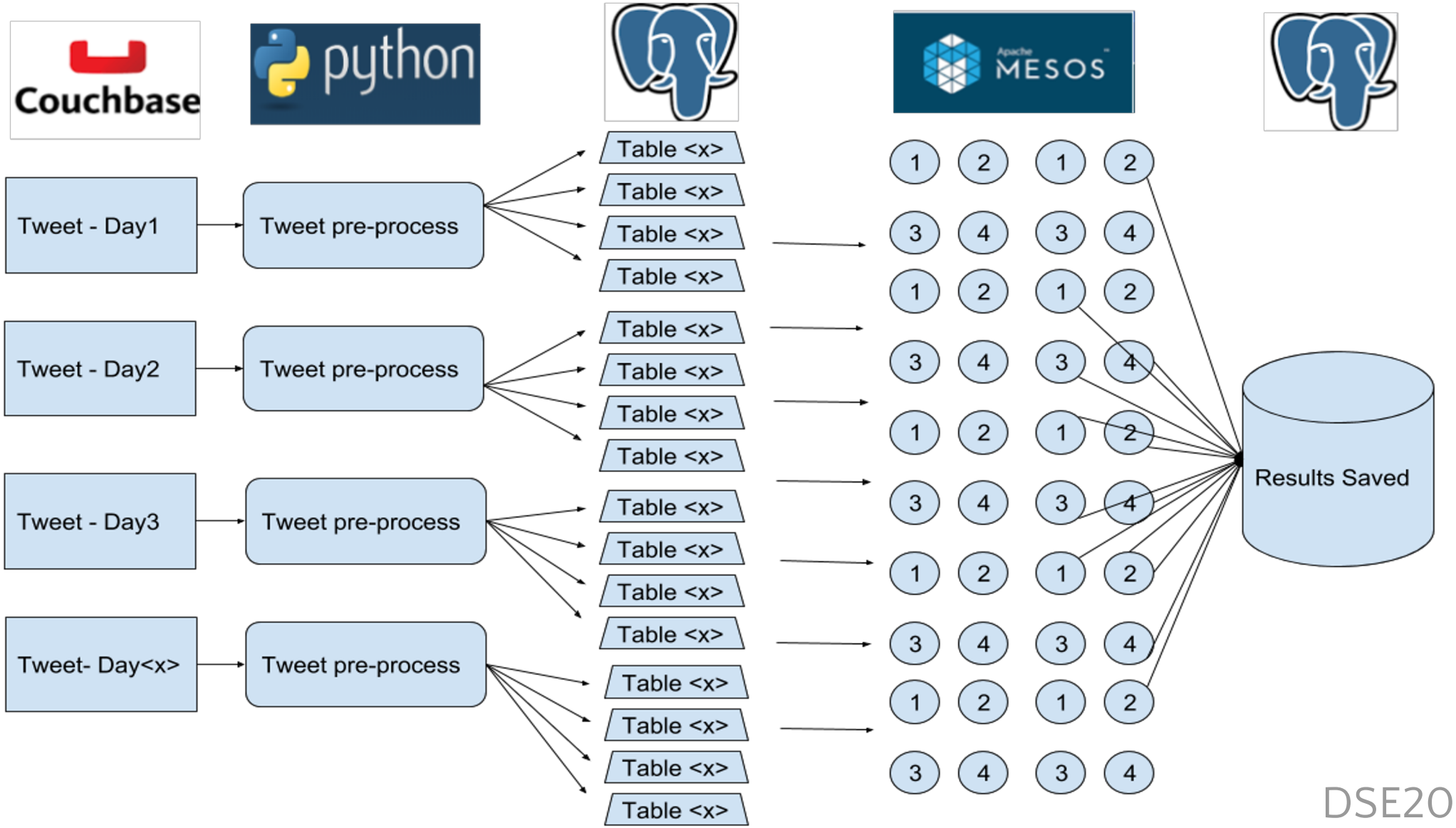
< Impl, ExecEng, Config >

Parameter	Input
Impl	Store News/Tweet Groupings
ExecEng	Python
Config	Apache Mesos Cluster

Data Flow Factors

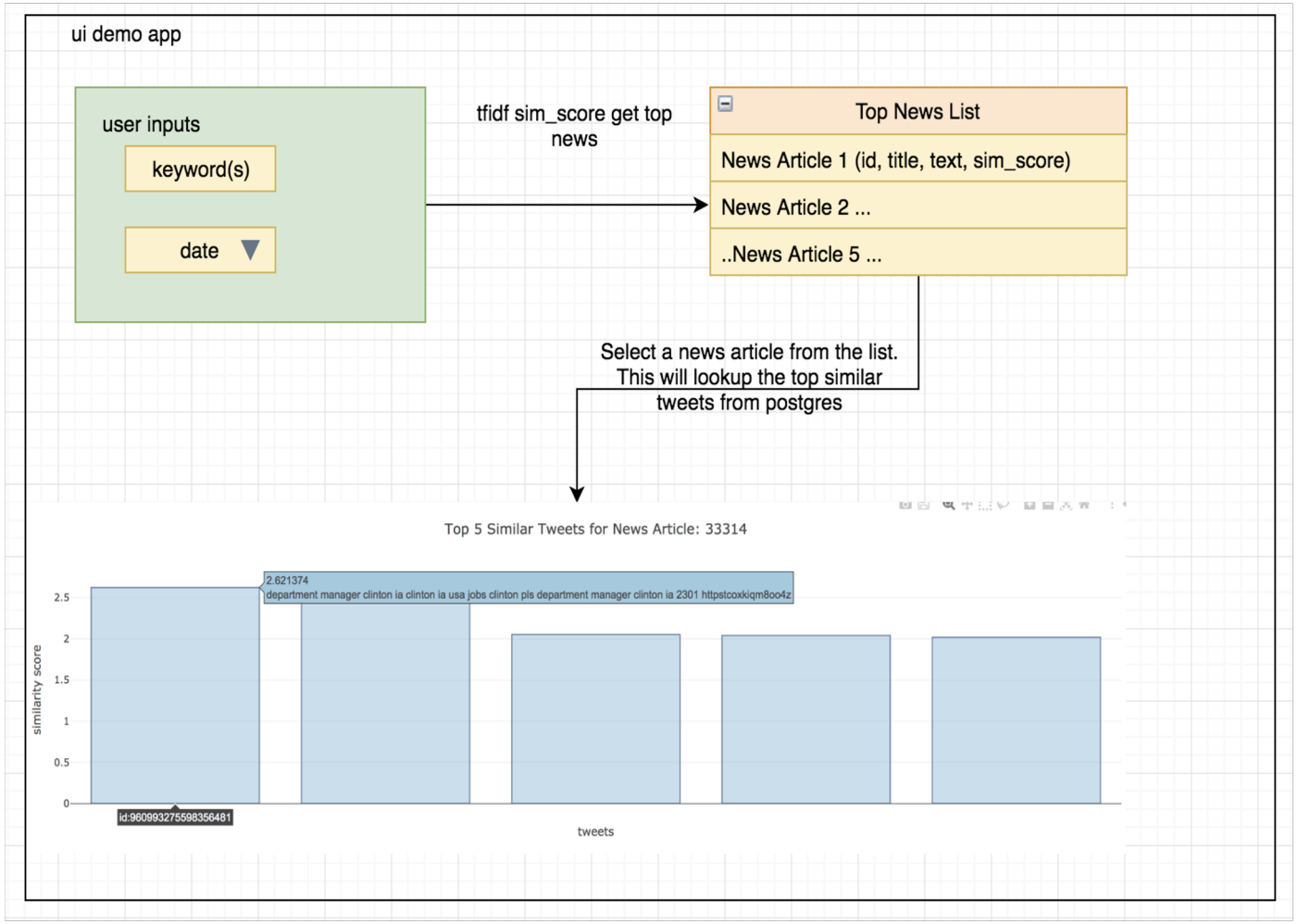
Factor	Description
Solution Type	Accurate
Application Domain	Batch
Execution Environment	Distributed

Notional View of Tweet Data Phases



UI Demo Workflow and UI

- allow a user to input keywords and a date to search for
- the search will take the keywords and apply the same tf_idf similarity alg. as we used before to get top news articles
- user can select a news article which will trigger a python workflow.
- The python workflow will pull the top similar tweets out of postgres and display in an interactive plot
- optimized with postgres indexes on tweetid



Future Improvements

- Utilizing Couchdb built in NLP for performance boost (Analyzers)
  - stemming-porter, apostrophe, to_lower, stop_tokens
- Using word vector libraries (ie. word2vec) for more complex embeddings
  - Build corpus specifically for tweets
- Create a golden data set in order to properly measure performance between different algorithms and embeddings
- Improve platform for increased divide-and-conquer and parallelism, removing increased storage points.