CS410 - Text Information Systems

News Source Sentiment Analysis & Topic Modeling using Twitter

{mmhorn2, peterwj2, dmorri25, lolowo2}@illinois.edu

Media is evolving at a quick pace, and many news sources are going beyond the traditional website. News now circulates across social media platforms, including Twitter. Our application retrieves recent tweets from the official Twitter accounts of prominent news sources. From the tweets, the application gleans insight into prevalent topics related to a user's search query. Once the user provides a query, sentiment analysis is performed using the TextBlob python package. Sentiment data is then gathered for each news source, and the user is equipped with information that will help them think more critically about tweets. The results reveal that the application can use topic modeling to help inform a user of current events, and the overall sentiment scores can help a user discover which news source's tweets align with their values or preferences.

Table of Contents

I. Front-End: Michaela and Peter

Our team used Flask to implement the front-end of the application. The user of the application first encounters the home page. All of screens of the user interface (UI) are implemented using Flask HTML templates. The main tool provided to the user on the home page is a search bar. The search bar is a text input field that accepts a user's query and sends the query to the back-end. The search bar is accompanied by the "Recommended Search Topics" section that includes buttons with topics derived from corpus *C* using the Probabilistic Latent Semantic Analysis (PLSA) model. Each time the home page is refreshed, the backend returns a new set of topics *T* with |T|=7. After each search, new documents are appended onto *C.* Terms related to queries become increasingly likely to appear in set *T* after each search. The home page also features a button to prune *C* if |C| becomes too large and hinders performance. Once the user sends the query and specifies news sources, the application retrieves tweets from Twitter and reveals much about the query on the Results page.

For one, the Results page displays each of the tweets along with the sentiment for each tweet and a subjectivity score for each tweet. The search field allows the user to filter by sentiment by searching the term "neutral", "positive", or "negative". A subjectivity score ranging between 0 and 1 is also available for each tweet. A subjectivity score above 0.35 is rendered in red to signal that the tweet contains substantial bias. Below the tweet table is a section that displays aggregate sentiment scores and subjectivity scores for each of the news sources. The aggregate sentiment scores are separated into percentage values. For each news sources, a distribution over "positive", "negative", and "neutral" tweets is displayed.

II. Back-End: Da'Mon and Lookman

The back-end service begins by deriving topics from a sample corpus using the PLSA topic model. The PLSA topic model returns the top 7 topics from a random sample of documents (*N*=30). Defining a sample size prevents excessive runtimes as |C| increases. As *C* grows, the likelihood of topics related to old searches will increase as well, making it more difficult to view topics related to recent searches. The delete_search_topics() function is implemented and reduces the corpus to |C|=30, a small enough number to maintain runtime performance levels, and large enough to minimize background words from being selected as topics. Once a user selects from the topics displayed or enters a custom query in the search bar, functions run in sequence to provide the user with the data detailed in the Front-End section.

The Twitter Client class uses the query data to query the Twitter API. The results from the Twitter API are parsed, tweet text is extracted, links are filtered out of the text, and metadata is filtered out as well, as it is not required in this use case. The TextBlob python package is invoked and performs sentiment analysis on each tweet. Magnitude and direction of the polarity measure are returned from TextBlob, which are later used by the front-end to display information on the Results page, detailed in Section 1. The tweets continue to be processed, and each tweet is sent to the tweet_refinery() function. Within said function, twitter handles (the unique twitter id beginning with the symbol '@') are removed, and words that have recently been searched are removed from the document to prevent overfitting on query terms (this begins after the 2nd search). Afterward, the Gensim python package is invoked to remove stop words, punctuations, non-alphanumeric characters, and words with less than 4 characters. If the document still contains words, the program appends the refined document onto the corpus. Once the document is updated and the user is viewing the results on the Results page, the user may click the logo at the top of the page to return to the Home page. Finally, the PLSA function shuffles the document order within the Corpus, collects a new sample (N=30), and a user can repeat this process as desired. (see Figure 1).

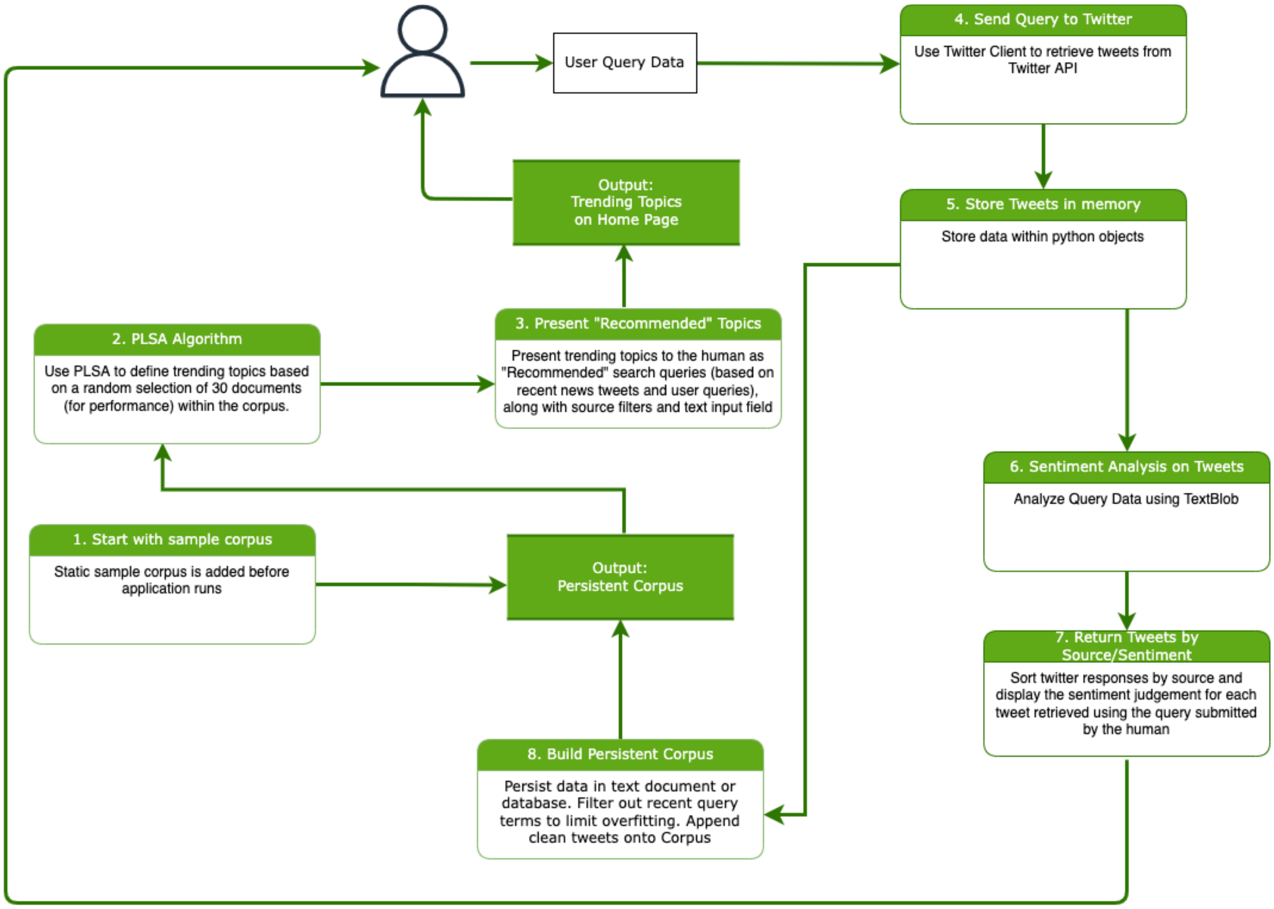
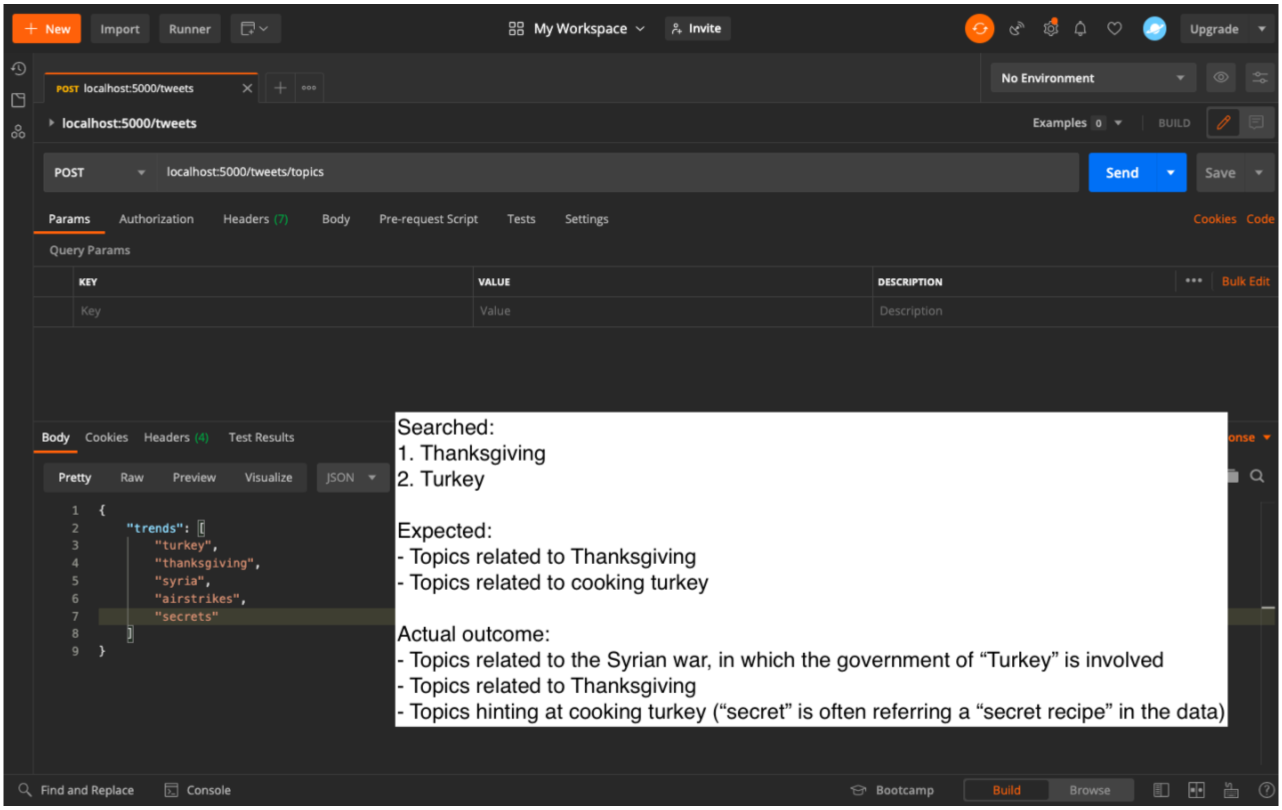

User Query Data
4. Send Query to Twitter
Use Twitter Client to retrieve tweets from Twitter API
Output: Trending Topics on Home Page
5. Store Tweets in memory
Store data within python objects
2. PLSA Algorithm
Use PLSA to define trending topics based on a random selection of 30 documents (for performance) within the corpus.
3. Present "Recommended" Topics
Present trending topics to the human as "Recommended" search queries (based on recent news tweets and user queries), along with source filters and text input field
6. Sentiment Analysis on Tweets
Analyze Query Data using TextBlob
1. Start with sample corpus
Static sample corpus is added before application runs
Output: Persistent Corpus
7. Return Tweets by Source/Sentiment
Sort twitter responses by source and display the sentiment judgement for each tweet retrieved using the query submitted by the human
8. Build Persistent Corpus
Persist data in text document or database. Filter out recent query terms to limit overfitting. Append clean tweets onto Corpus

Figure 1

III. Conclusion

Creating this project was engaging and interesting, and the results actually useful. The application equips a user to intentionally consume information, as opposed to impulsively consuming the content that the social media website chooses to show them. This empowers said user to be more mindful while using social media. This is achieved without sacrificing the connectedness to the real world. The application is connected to the Twitter accounts of official news sources, meaning that it can legitimately inform a user of current topics that are being talked about in public forums (see figures 2, 3, and 4). Our evaluation of the application confirmed functional consistency, with one caveat that I will explain. For a given query ("biden"), one news source's (Fox News) tweets consistently ranked as the 1st or 2nd least positive in aggregate positive sentiment, while another source (CBS News) consistently had the 1st or 2nd highest positive sentiment on the given query. Therefore, if a user prefers to read positive news about Joe Biden, the user can subscribe to CBS news as opposed to Fox News. The caveat to this is that the data was only checked three times between November 9, 2022, and December 7, 2022. This is a small sample size (N=3), and some of the tweets could be the same for each sample, which would cause the values to be consistent. Future iterations of this application may incorporate the metadata that the Twitter API offers. This would allow a user to search for tweets within a given country, or during a given timeframe, and Contextual PLSA can be performed on the results. This would further equip the user to be more mindful about their browsing.



POST localhost:5000/tweets/topics

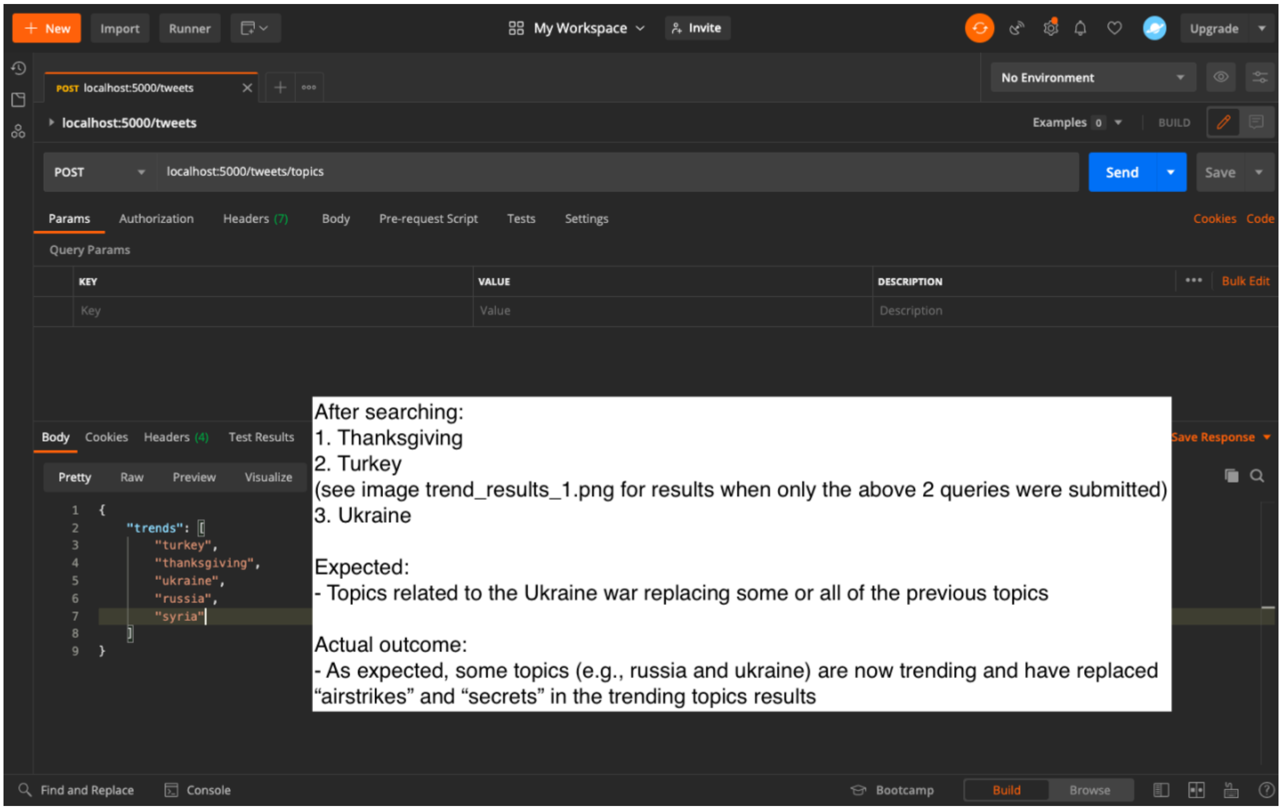
```
{
    "trends": [
        "turkey",
        "thanksgiving",
        "syria",
        "airstrikes",
        "secrets"
    ]
}
```

Searched:
1. Thanksgiving
2. Turkey

Expected:
- Topics related to Thanksgiving
- Topics related to cooking turkey

Actual outcome:
- Topics related to the Syrian war, in which the government of "Turkey" is involved
- Topics related to Thanksgiving
- Topics hinting at cooking turkey ("secret" is often referring a "secret recipe" in the data)

Figure 2 (prior to preventing overfitting)



POST localhost:5000/tweets/topics

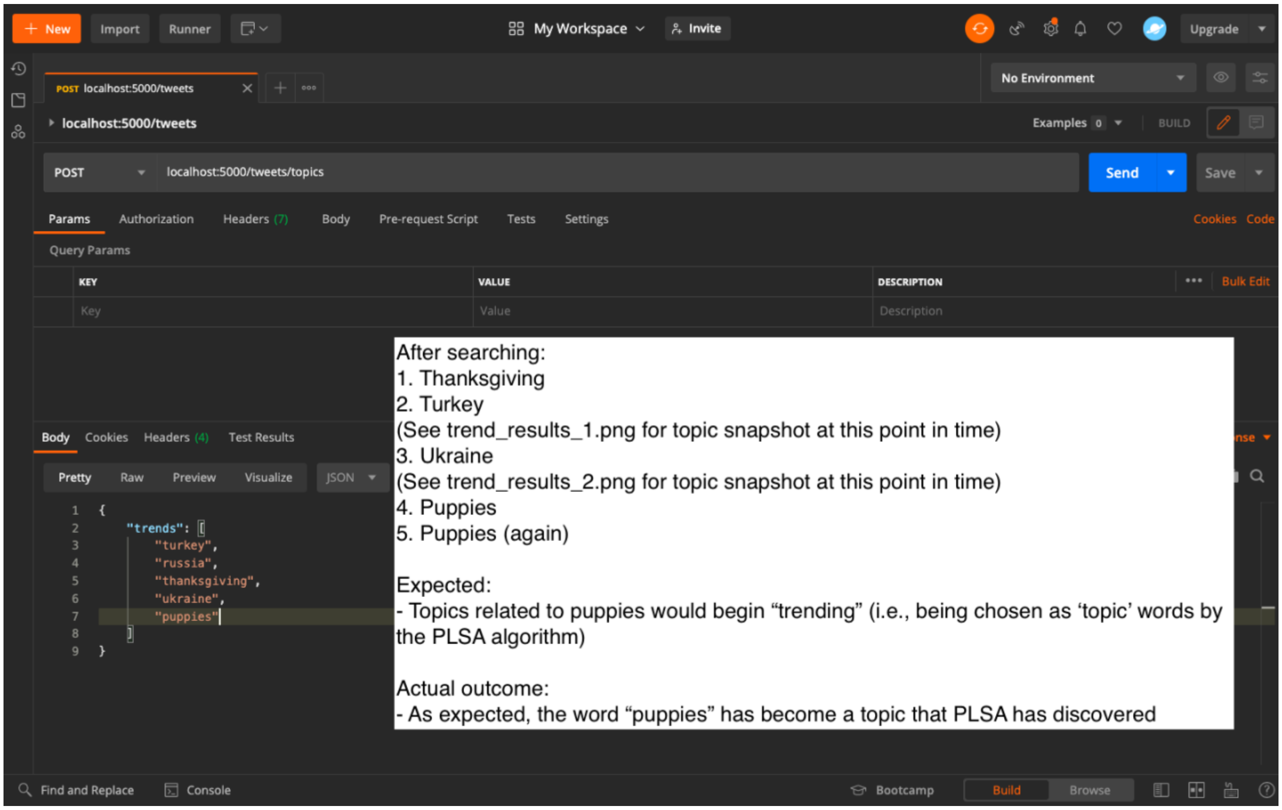
```
{
    "trends": [
        "turkey",
        "thanksgiving",
        "ukraine",
        "russia",
        "syria"
    ]
}
```

After searching:
1. Thanksgiving
2. Turkey
(see image trend_results_1.png for results when only the above 2 queries were submitted)
3. Ukraine

Expected:
- Topics related to the Ukraine war replacing some or all of the previous topics

Actual outcome:
- As expected, some topics (e.g., russia and ukraine) are now trending and have replaced "airstrikes" and "secrets" in the trending topics results

Figure 3 (prior to preventing overfitting)



POST localhost:5000/tweets/topics

```
{
    "trends": [
        "turkey",
        "russia",
        "thanksgiving",
        "ukraine",
        "puppies"
    ]
}
```

After searching:
1. Thanksgiving
2. Turkey
(See trend_results_1.png for topic snapshot at this point in time)
3. Ukraine
(See trend_results_2.png for topic snapshot at this point in time)
4. Puppies
5. Puppies (again)

Expected:
- Topics related to puppies would begin “trending” (i.e., being chosen as ‘topic’ words by the PLSA algorithm)

Actual outcome:
- As expected, the word “puppies” has become a topic that PLSA has discovered

Figure 4 (prior to preventing overfitting)