News Sentiment Tracker: A Targeted Opinion Mining Interface

Andrew Wesson | Prashanth Rao

Motivation

More than **2.5 billion** people read online or print news articles on a near-daily basis. With such large reach and unparalleled scale, the effect of strongly positive or negative news coverage of a target can linger for *long periods* after a triggering event.

Ryan Lochte
"Lochtegate" (2016)

- **Millions** lost in endorsements
- 6-month swimming suspension

United Airlines
2017 Controversies

- **1 billion** dollar loss in share value
- CEO resigned over backlash

Liberia
Ebola (2014/16)

- **Multi-billion** dollar loss in GDP in 2015
- Lower investment in economy for years

Figure 1. Example implications of negative coverage

Goal

Build an intelligent, **automated system** that analyzes news content towards a target over time, and aggregates the sentiment of *specific mentions* of the target **to assist relevant personnel in decision-making**.

Dataset

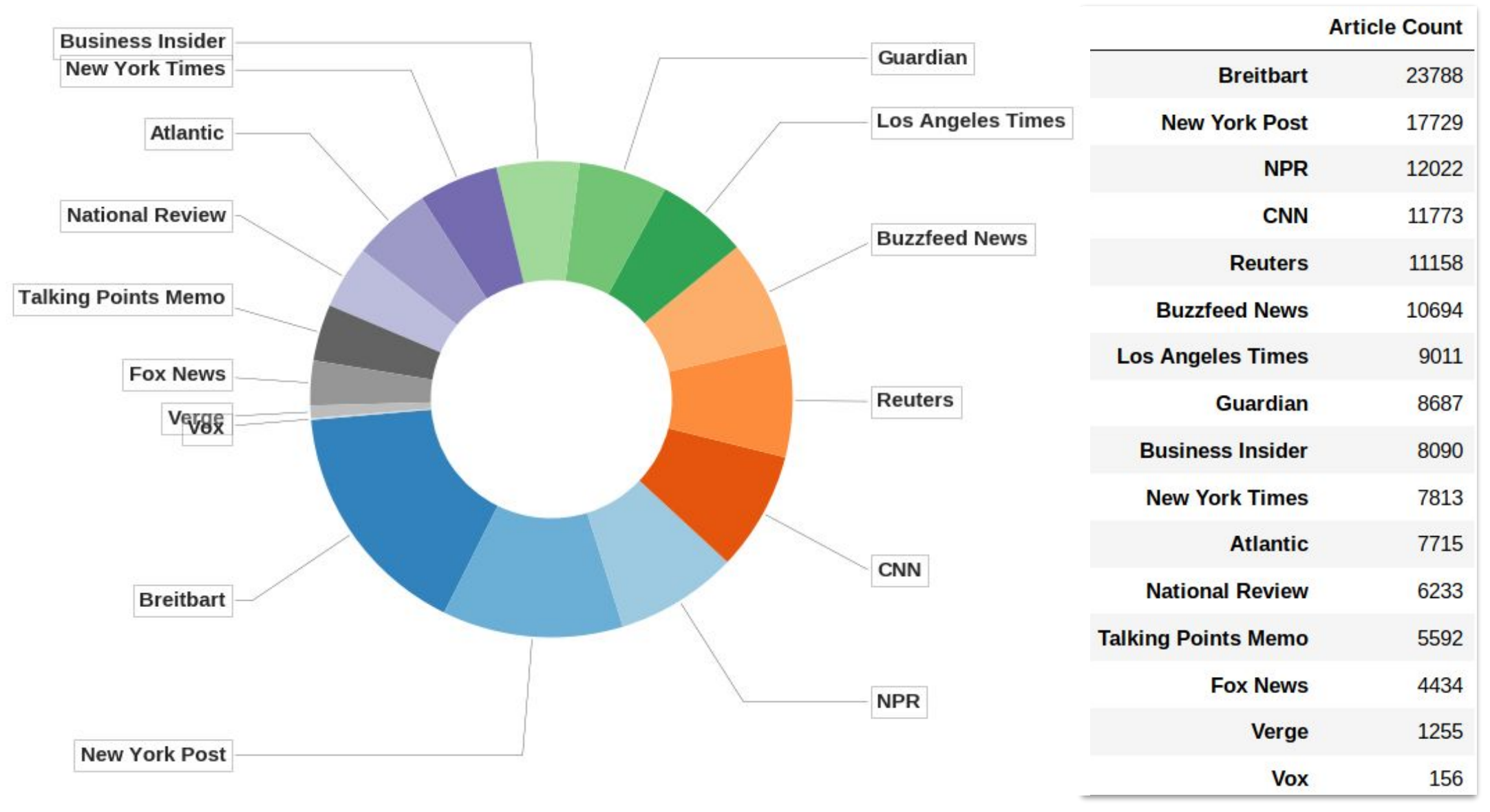
To solve this problem effectively, we require a news article dataset that has the following attributes:

- Broad variety in news topics
- Long term (i.e. contains articles from multiple years)
- Diverse range of publications

"All the News" Dataset

This is a public dataset hosted on Components[1] that has **143,000** news articles from sixteen U.S. publications.

[1] https://components.one/datasets/all-the-news-articles-dataset/

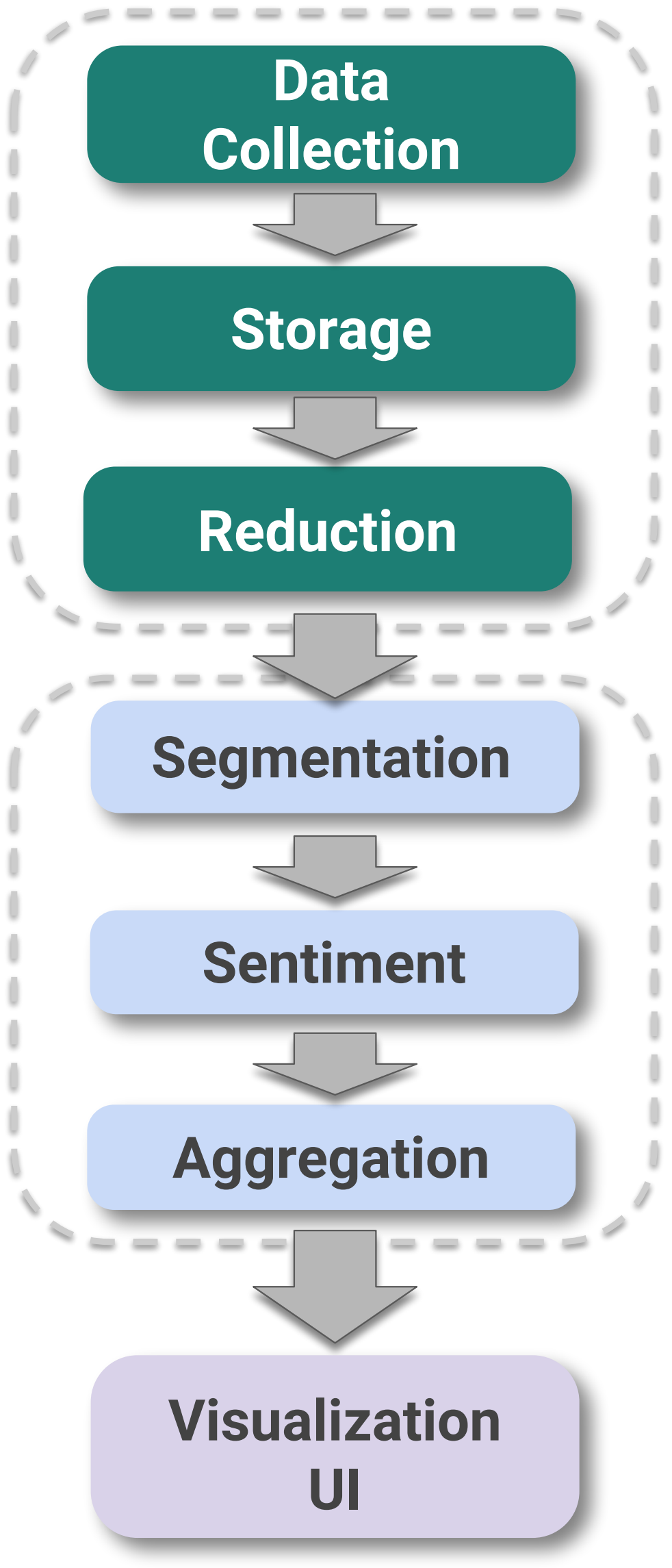


	Article Count
Breitbart	23788
New York Post	17729
NPR	12022
CNN	11773
Reuters	11158
Buzzfeed News	10694
Los Angeles Times	9011
Guardian	8687
Business Insider	8090
New York Times	7813
Atlantic	7715
National Review	6233
Talking Points Memo	5592
Fox News	4434
Verge	1255
Vox	156

Figure 2. Breakdown of news article distribution by publication

- Most articles are in the period between ***2014*** to ***2017***
- The median article length is between 300 and 1,500 words.

Methodology



Data Pipeline

- Scraper + News API (not implemented yet)
- Indexed storage
- Fast retrieval of relevant news article indices
- Boolean logic for combining search terms
- Tools: ***PySpark, Lark***

NLP Pipeline

- Extract relevant sentences from content
- Sentiment analysis on a per-sentence basis
- Aggregate sentiment to track large-scale shifts
- Tools: ***spaCy, fastText, Flair***

Dashboard

- Deploy sentiment time series as a web app
- Tools: ***Plotly Dash, Heroku***

Data Indexing and Retrieval

To facilitate the easy retrieval of relevant articles for any given topic, we built a **Boolean search** system using Apache Spark. This system allows us to efficiently select articles based on the presence or frequency of keywords. These search terms can be combined using **and, or,** and **minus operators**.

Task	Query
Identify articles about Samsung's Galaxy Note phone series	`Samsung & Galaxy & Note`
Distinguish between the country China and the *material* china	`China - (china | porcelain)`
Find articles mentioning Martin Shkreli multiple times	`Shkreli > 5`

To query using this approach, we add each article to an **inverted index** that stores, for each distinct word in the dataset, the IDs of the documents in which that word appears, and the number of times it occurs.

The index entries for each term present in the query are retrieved. Then, the query is converted to a predicate function that combines the entries for the different terms for each article in the specified manner. This function determines whether the article will be included in the result.

Sentiment Analysis

We implemented and compared three separate sentiment analysis pipelines, two of which apply machine learning.

TextBlob	*fastText*	*Flair*
Rule-based	Classical word embeddings	Contextual string embeddings
CPU	CPU (multi-threaded)	GPU (deep learning)

Fine-grained Sentiment Training

We chose the Yelp 5-class review dataset to train our fastText and Flair models.

- fastText is *really* fast - More than a million text samples were used for training the model in under **3 minutes**
- Flair takes a **few hours** to train with around 35,000 samples on a GPU, since it uses a PyTorch-based language model
- Both models were made to predict mean sentiment in a continuous scale of [-1, 1], on the same test cases

Custom Scoring Metric

To judge the sentiment polarity of the extracted text, we score the results based on a **per-sentence** evaluation of sentiment. We define the ***mean score*** and ***mean deviation*** metrics for each article.

'Monday, Michelle Obama and her daughters, Sasha and Malia, begin their lightning tour of Liberia and Morocco to promote the Let Girls Learn initiative. 'Massa David, a tall, lanky 15-year-old in the sixth grade, would like to take Obama on a tour to show her some of the problems with Liberia's infrastructure.' 'Poverty, sexual exploitation both in and outside Liberian schools and teen pregnancy are major factors that stop girls from completing their education in Liberia.'

Scoring each sentence individually and comparing it with the deviation in score allows us to identify mixed sentiment coverage, which we then plot as a time series.

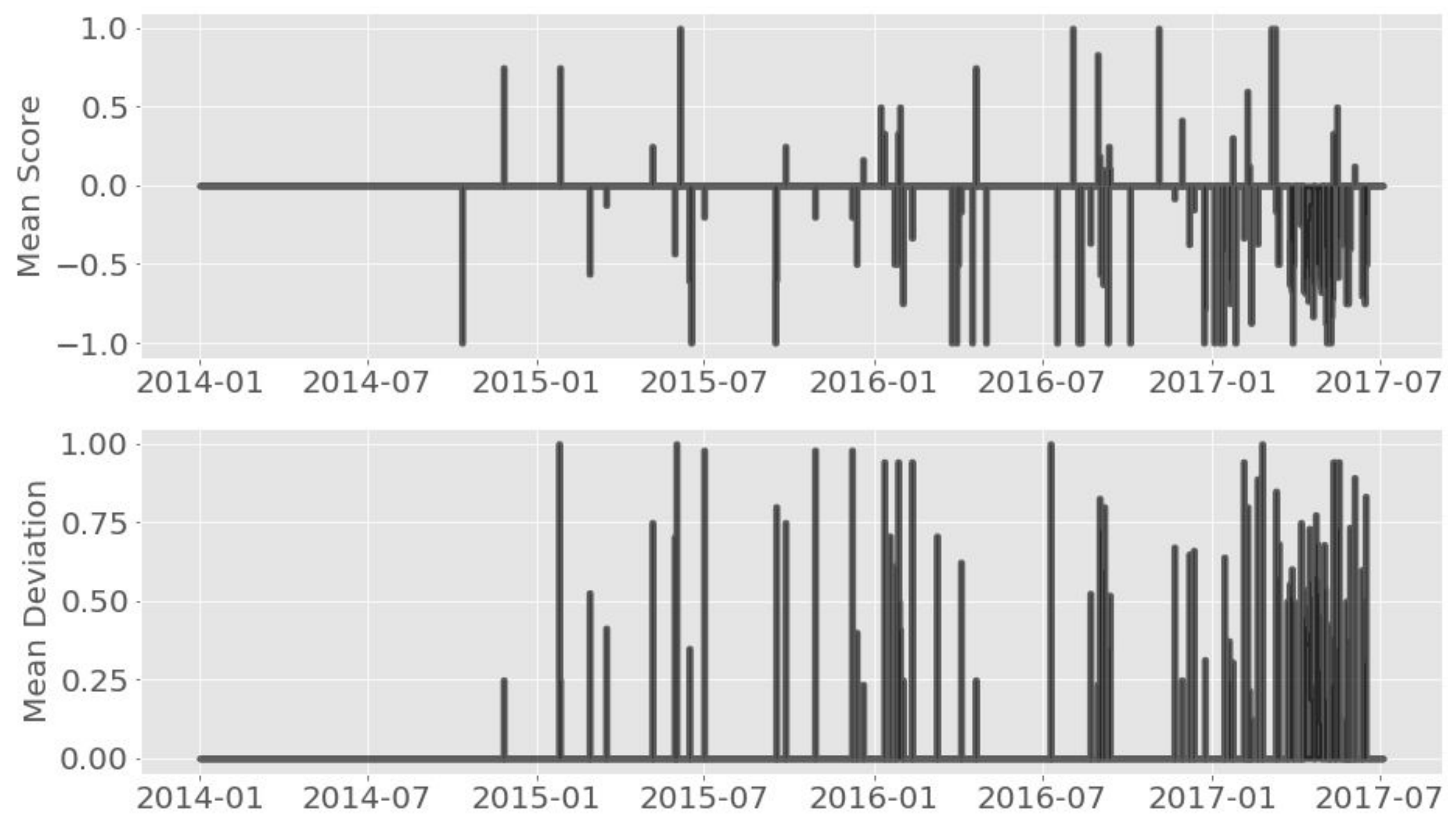


Figure 3. Mean score and deviation time series for the query "United Airlines"

Discussion

With the appropriate level of sophistication in our language model for sentiment analysis, more accurate trends can be matched with our expectations from reality. In our studies, simple rule-based approaches like TextBlob fail to capture negative sentiment trends, for example, the 2017 United Airlines scandals.

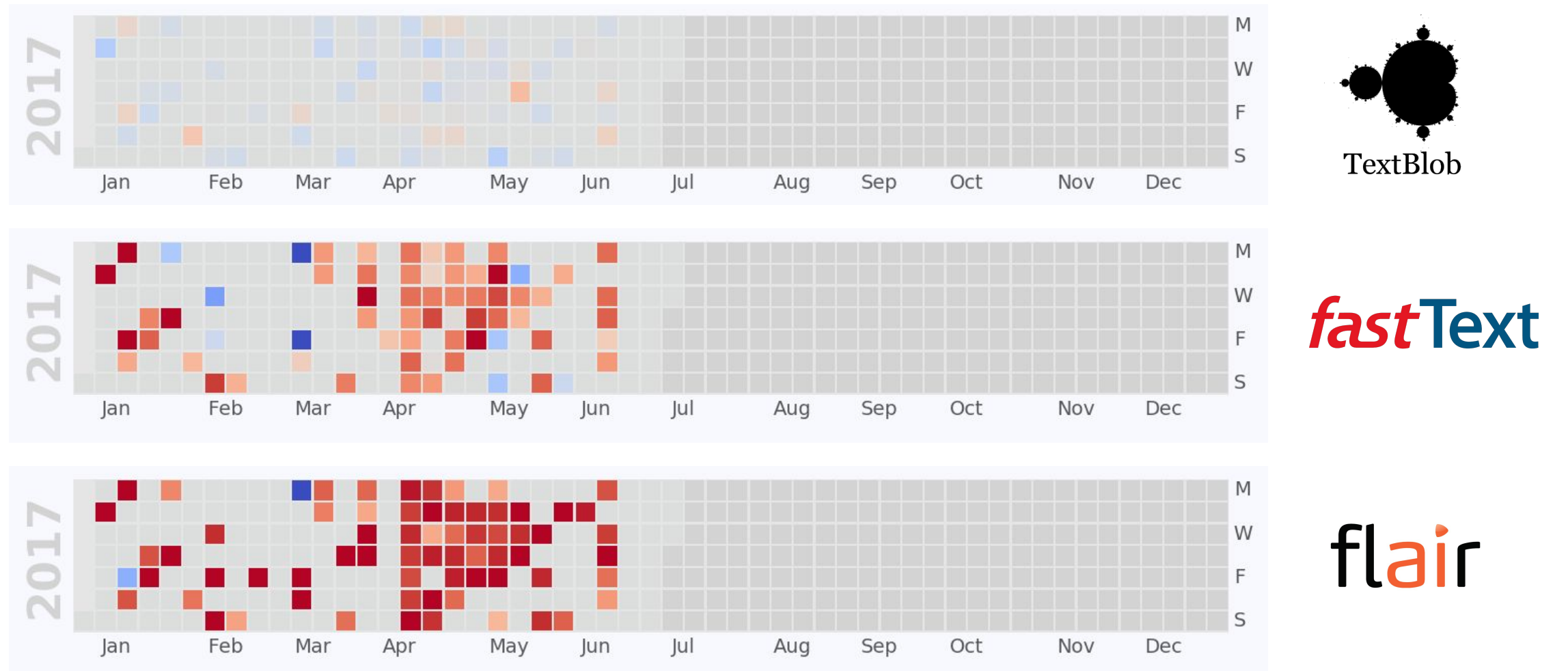


Figure 4. Calendar heat map using 3 sentiment models for "United Airlines"

It is clear that the bi-LSTM language model used by Flair better captures the negative coverage towards the target through its contextual representations. While fastText also considers words in the context of its neighbours, it only uses bigrams, which is not as powerful as a neural language model.

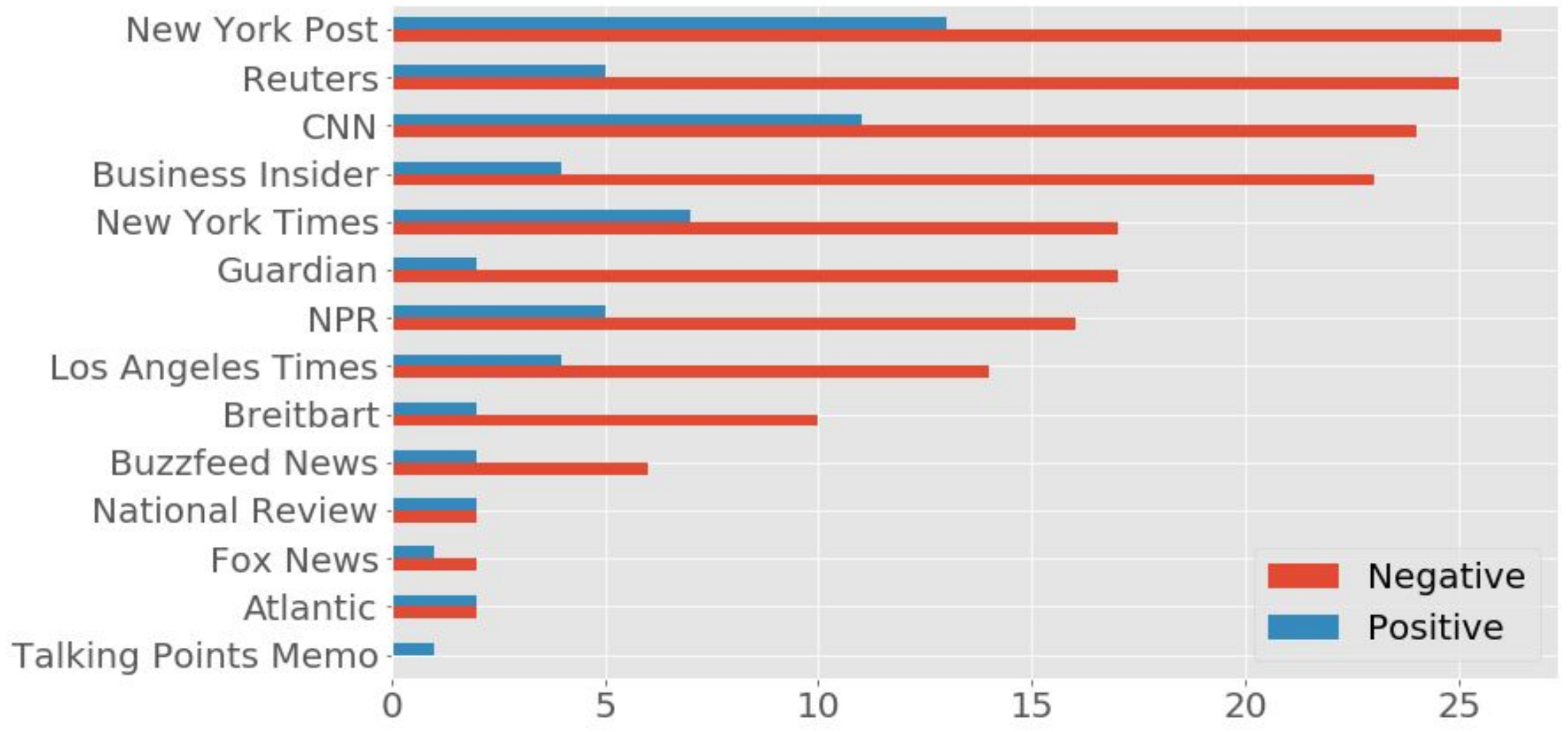


Figure 5. Breakdown per publication for the query "United Airlines"

We also show the per-publication breakdown of positive and negative content (*not just headlines*) towards a target.

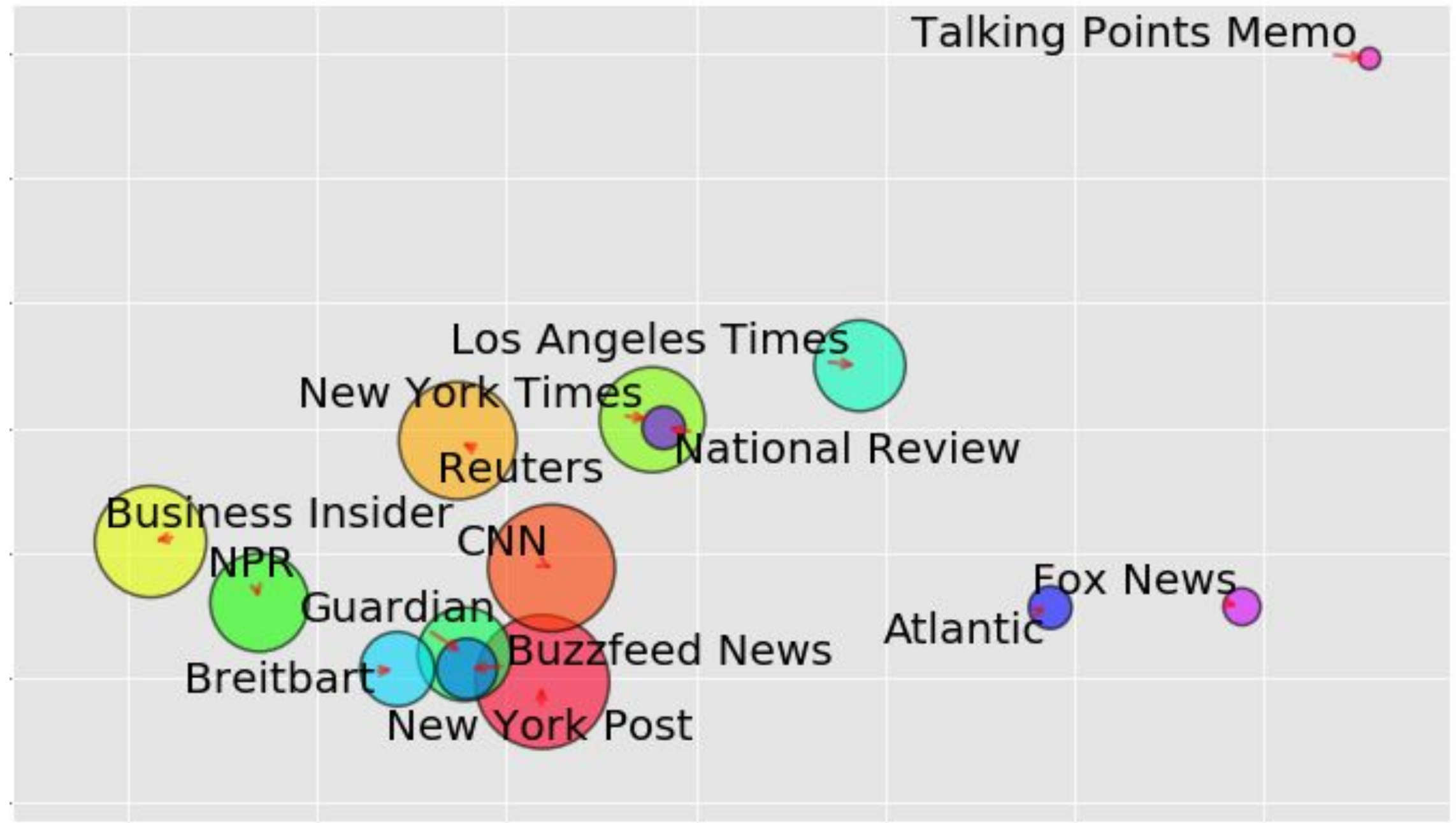


Figure 6. Mean cosine distances per publication for the query "United Airlines"

The extracted content is then converted to a *Term-Frequency Inverse Document Frequency* (TF-IDF) matrix from which we compute cosine distances between each article. To visualize this, we plot a *Multi-Dimensional Scaling* (MDS) plot to reduce the multi-dimensional cosine distances to two dimensions.

Conclusions

Our NLP-based product was conceptualized to be:

- **Flexible**: Applicable to people, organizations, products, etc.
- **Scalable**: Designed to process large news datasets
- **Meaningful**: Produces trends that match reality

Future Improvements

- Update our indexing scheme to reduce file size
- Test more complex boolean search queries to narrow down on more specific topics and events of interest
- Improve the similarity detection measure to use "*document embedding distance*" rather than cosine distance

See the web app: https://nlp-733-dash.herokuapp.com