Making Sense of Misinformation At Scale

Creating a Data Pipeline to Automate Topic Modeling from the Crowd

Ethan Anderson, Jesse Chamberlin, Lucy Eun, Evan Frawley

Advisors: Amirah Majid, Jevin West, Vinny Green

Abstract

The web is effective at disseminating information at the click of a button, and fact-checking organizations have a dearth of resources to keep up in our viral age. Prioritization of this information will allow for more efficient and accurate topic selection. We discuss an implementation of an automated NLP topic-modeling system at Snopes.com, “the oldest and largest fact-checking site on the internet”. This is accomplished through a data pipeline which harnesses the power of the crowd. Crowdsourcing is used to collect user reports of pages through a custom web client. HTML pages of reports are then scraped and parsed into meaningful clusters for the Snopes.com reporting staff to act on. Real-time topic modeling provides metrics for prioritization, and effective allocation of resources. Our data pipeline minimizes noise, and reduces the amount of manual curation required by editors and journalists at Snopes, which enables them to reallocate their limited resources to debunking more rumors.

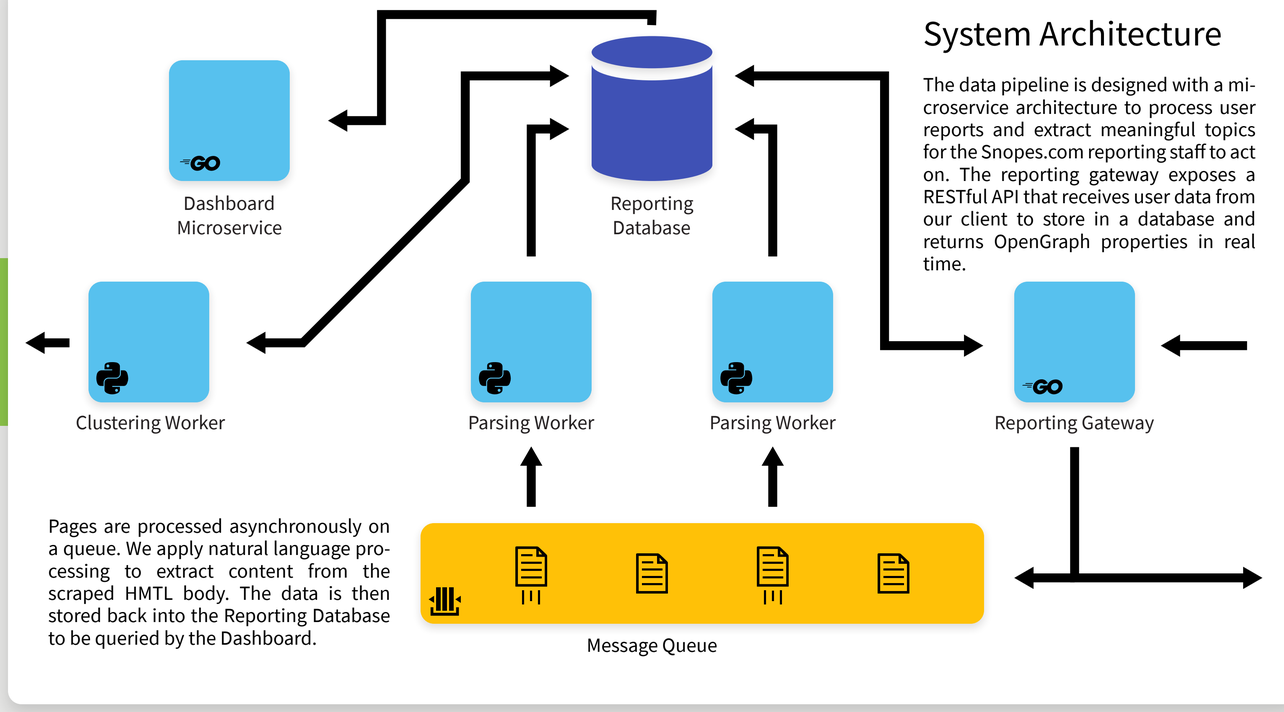
how do rumors get debunked?

Users are able to report leads to Snopes.com via our progressive web app reporting client at report.snopes.io.

A report consists of one or more pages (web content with a unique URL) associated with –context or commentary

System Architecture

The data pipeline is designed with a microservice architecture to process user reports and extract meaningful topics for the Snopes.com reporting staff to act on. The reporting gateway exposes a RESTful API that receives user data from our client to store in a database and returns OpenGraph properties in real time.



Pages are processed asynchronously on a queue. We apply natural language processing to extract content from the scraped HMTL body. The data is then stored back into the Reporting Database to be queried by the Dashboard.

HTML pages are scraped from user-submitted URLs and archived with the Wayback Machine upon submission in an effort to preserve the content before it can be altered or taken down.

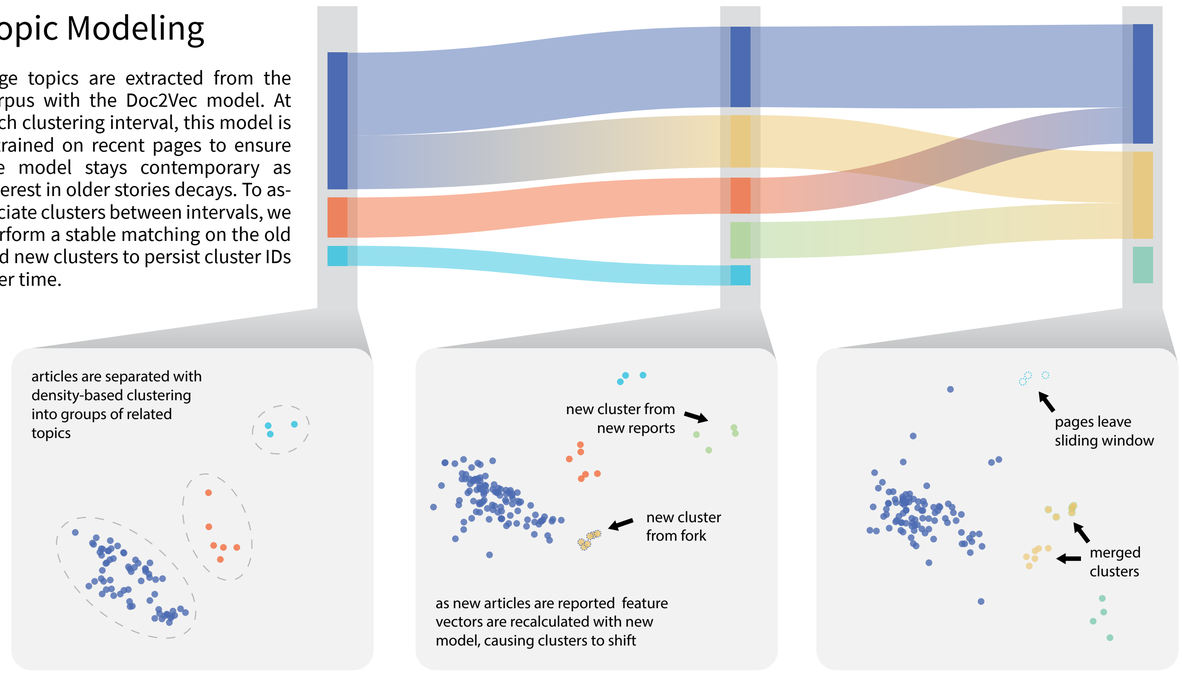
Clustering

Document clustering is an unsupervised learning technique for grouping similar documents. Our corpus of user-submitted conent evolves over time as new topics emerge and old ones fade away.

To facilitate real-time monitoring of claims, we temporarily assign new reports to existing clusters as soon as they have been parsed. At a set interval, the topic model is re-trained on a sliding window of the most recent pages. Since a page stays in the window for multiple intervals, its position relative to other pages evolves with the model. At each interval, clusters are recomputed. Cluster history is stored for each page to track forking and merging of clusters over time.

Topic Modeling

Page topics are extracted from the corpus with the Doc2Vec model. At each clustering interval, this model is retrained on recent pages to ensure the model stays contemporary as interest in older stories decays. To associate clusters between intervals, we perform a stable matching on the old and new clusters to persist cluster IDs over time.



Scoring Metric

Once pages are assigned into clusters, we compute a triage score for each cluster to enable sorting and comparison between them. The triage score is an aggregate of the individual page scores and inter-page relationships. Pages that are reported more than once have a higher score, as well as content rehosted across multiple domains that appears as multiple pages. The age of a page as measured by time since report is also factored in. The score of a page decays exponentially over time to favor clusters with more recent pages.

Pages can also be retroactively clustered into inactive clusters. Density based clustering can leave pages unclustered, so as a post-processing step, unclustered articles are compared against old clusters to search for one that matches. This can "revive" an old clusters if enough unclustered articles get assigned to it, in the scenarios where a dormant story resurfaces after an extended silence.

Conclusion

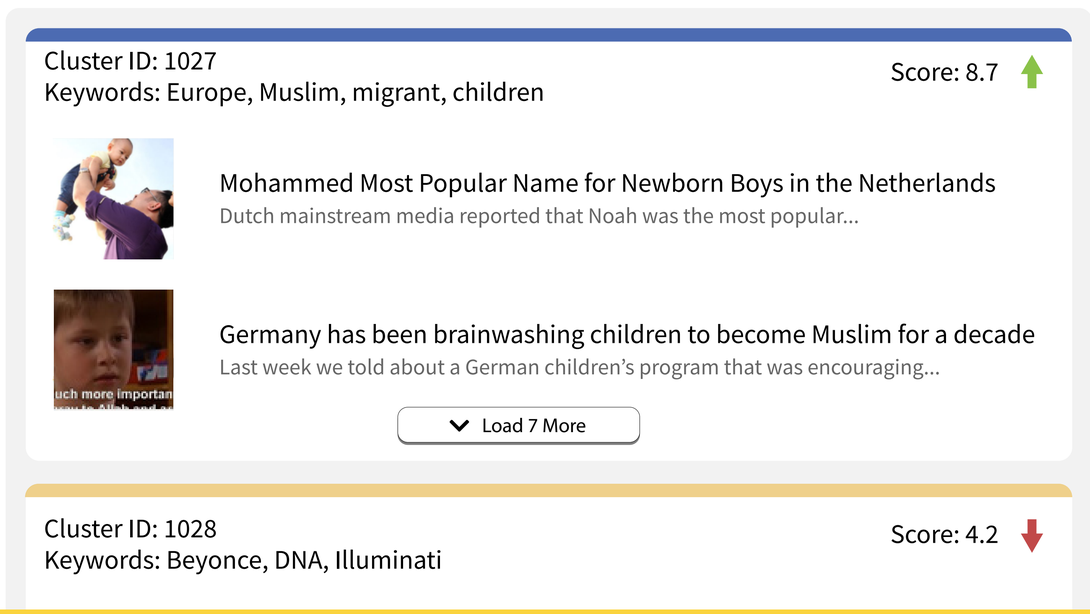
As a solution to fight misinformation in a world of viral content, we propose a system to present an accurate prioritization of information that fact-checking organizations can act on. Our data pipeline for automated topic modeling harnesses the crowd to get an acurate representation of topic importance, and minimizes the noise of misinformation by prioritizing the most important topics.

Future Work

In the future, we plan to cross-reference social media websites like Facebook, Twitter and Reddit to spread the triage metric across multiple platforms make our score more robust, and to make it more difficult for bad actors to game the system. We also need to experiment with more sophisticated clustering models that take non-text data into account. Including page domain, publication date, author names, and comments in the model may prove more effective at isolating clusters than primary content alone. As the system becomes more widely adopted, it will also be neccesary to (say something about scaling and filtering bad actors

Reporting & Triage

The dashboard displays the clusters as a feed to the Snopes.com reporting staff, allowing them to more easily monitor and address misinformation. Each cluster is listed with its keywords, page metadata, and aggregated "triage score". The "triage score" is derived using a proprietary aggregation of cluster density, temporal data, volume of associated reports. The goal of the dashboard is to identify and communicate structure, trends, and anomalies in the collected user reports. Consequentially, the dashboard increases the efficiency of the Snopes.com reporting staff that author fact-checking articles.



“A lie can travel halfway around the world while the truth Is putting on its shoes.”

- misattributed to Mark Twain

Snopes DataLab Information School UNIVERSITY of WASHINGTON