CrossCheck Technical Structure

Overview

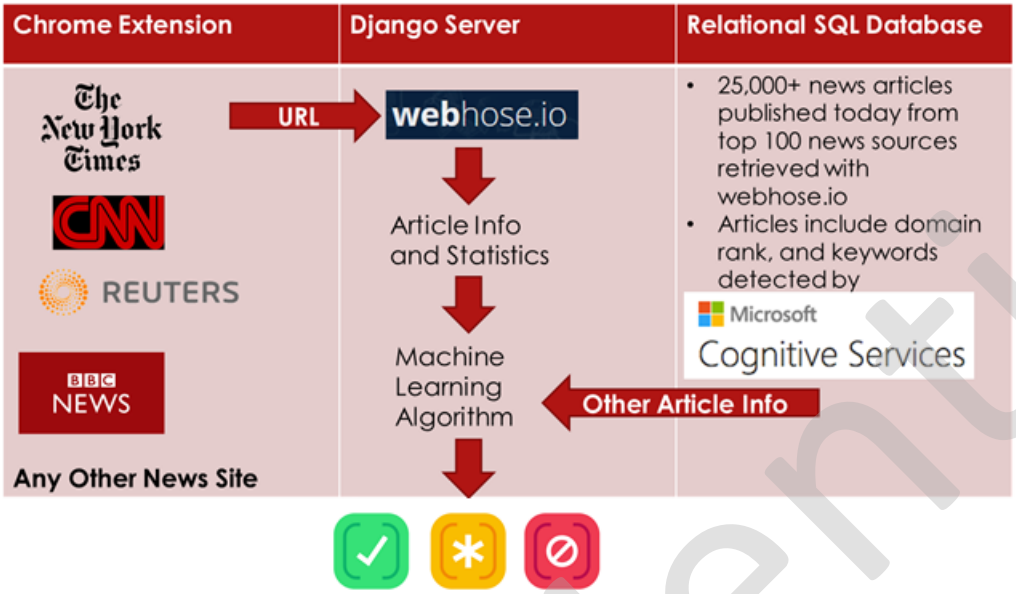

Figure 1. **Visual overview.** Our product has three components, a user client (either chrome extension, and later web app and mobile app, and finally API), a Django Server, and a SQL Server backend.

Relational SQL Database & Apache Tomcat Server

Every 5 minutes, our script in Java queries all news articles published from the top news sources using webhose.io. These news articles including a host of ancillary information (see Database Schema Design) and the information is efficiently saved to the Microsoft SQL Server hosted on Azure. We use an asynchronous multithreaded approach to maximize efficiency and effective parallelization. The code runs as an Apache Tomcat Webapp on Microsoft's Azure server.

Django Server

The Django Server is the brain of the backend. Processing user requests as a URL, retrieving information from webhose.io corresponding to that URL and running the information through the algorithm. The backend can easily scale to support millions of clients and interfaces seamlessly with our SQL Server. The Django server is also responsible for periodically querying the SQL Server and pre-computing the clusters/category that are used in the algorithm.

User Clients

Chrome Extension

Our first client is a chrome extension that works like AdBlock extensions. As users browse the internet, the extension checks the site they are on. If the site is considered fake news, the user is alerted. A fancy visualization displays when clicked on. The chrome extension also keeps track of news websites that

users visit and verify to later be sold to advertisers and news organization (users can opt out and get premium feature for 10 USD per month)

Web and Mobile Apps

We are operating under the “lean startup” methodology so we want to get our initial chrome extension out there first. Once we do this, we will connect the chrome extension to a web app, a news aggregation website that marks articles as verified or not. The web app and mobile apps will use ML algorithms to keep track of user preferences. News companies can choose to be featured on the web app by paying a fee based on the number of users they would reach.

API

The final user client is a REST-like API. This will allow major clients such as google and facebook to funnel millions of calls to the API for massive verification of news articles.

Database Schema Design

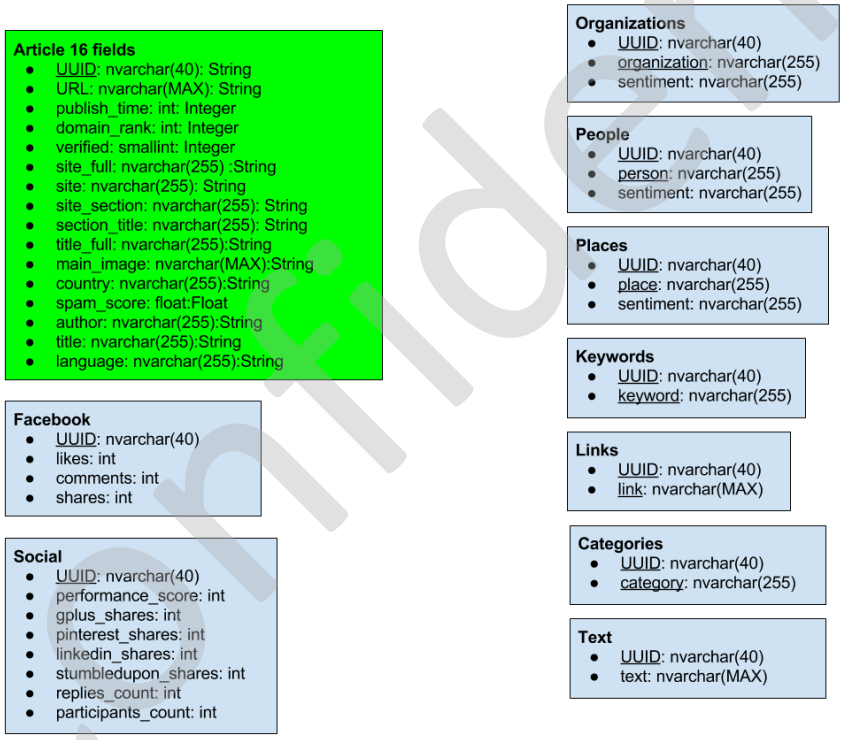

Figure 2. Database Schema Design. We host our database on Azure and use Microsoft SQL Server. We have an efficient database schema that allows us to do quick queries and joins given certain selection criteria.

Algorithm Overview

Step 1: Spam Filtering

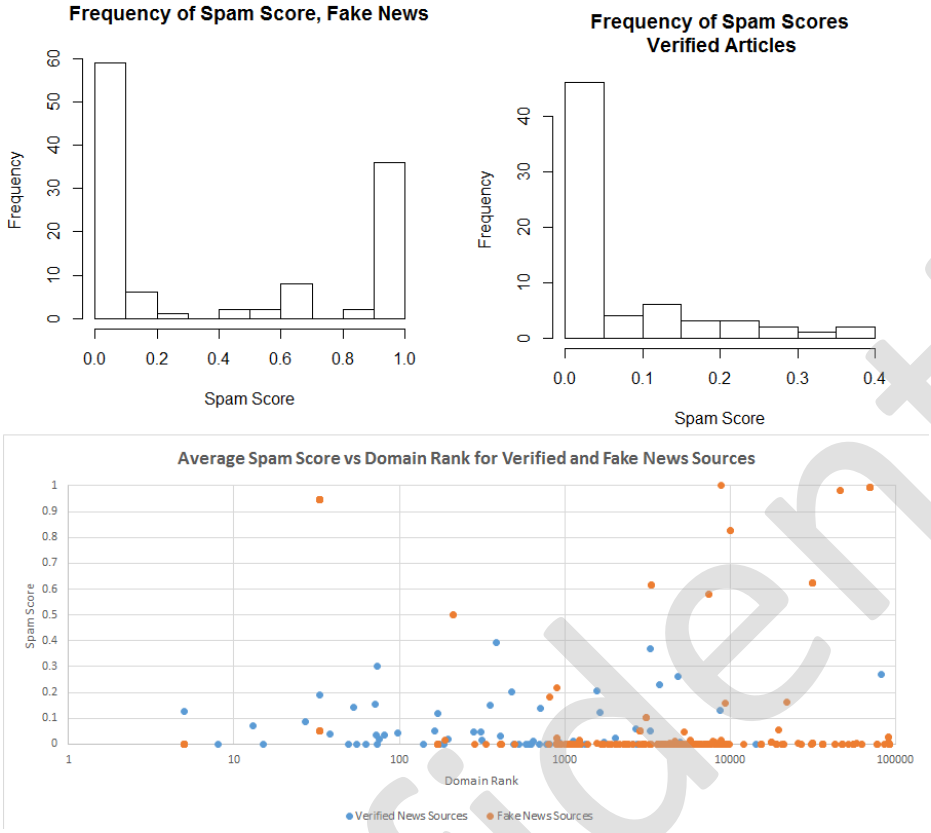

Figure 3. Spam Index Separation in Fake and Verified Sources. Spam scores are calculated by webhose.io using Stanford's NLP library and are clearly distributed differently for fake news sources and verified articles. By simply setting a threshold of 0.42, 40% of fake news articles can be distributed. When we train a Support Vector Machine Model on Spam Score and Domain Rank, we are able to obtain an accuray of about **90%** for fake and verified news classification. Note that while this basic algorithm works, it is **flawed** as fake news sources will naturally have less traffic than real news sources. If a fake news source with a low spam score recieves a high domain rank, it could be marked as verified. Fortunately, there aren't many of these types of articles. The **CrossCheck Score** returned by the algorithm is 1-spam_score is spam_score > 0.42, the cutoff determined by our simple machine learning model. If spam_score <=0.42, we use dimensionality reduction and constant time bayesian agglomerate cluster membership check (see below).

Step 2: Dimensionality Reduction

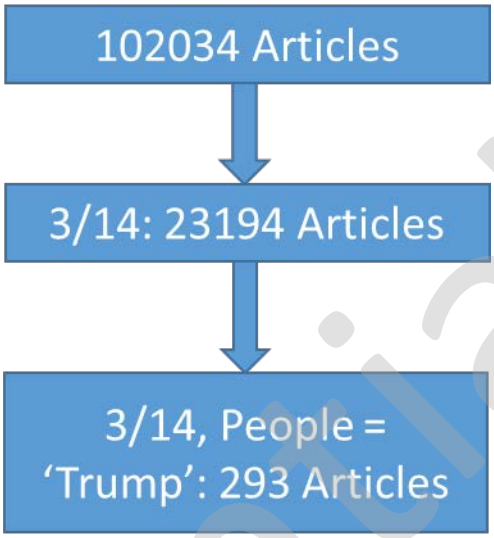

Figure 4. **Example Dimensionality Reduction Flow Chart.** With thousands of articles added to the database every day, we need a way to compare a reasonable number of articles for similarity. To perform this dimensionality reduction, we start by taking all articles published within a 24-hour period. Then we look at the categories, people, places, and organizations in an article as determined by webhose.io. For each of these entities, we select all the articles in the category. Given that the cardinality of the category is larger than some size, epsilon, we proceed with step 3.

Step 3: Constant Time Bayesian Agglomerate Cluster Membership Check

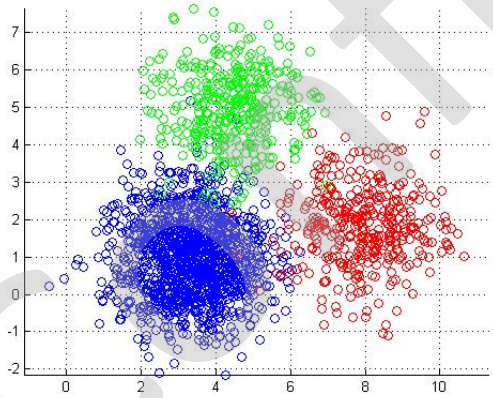

Figure 5. Clustering Visualization Example. *Note that this figure is not based on actual data, we have yet to map the multidimensional feature clustering to the Cartesian plane*. We are constantly pre-computing clusters of articles using a variety of metrics form webhose, Stanford NLP libraries, and others given an entity. A "cluster" of articles is articles that should be all about the same thing given an entity. We determine the **CrossCheck score**, the probability that a query article is part of an existing cluster of

articles using the following formula. Because we precompute the clusters in real time as we scrape more data, this probability is an amortized constant time operation.

$$\mathbb{P}(existing\_cluster|clusters, entity) \propto L(existingclusters|clusters, entity) \cdot \mathbb{P}(Clusters|entity)$$
$$\mathbb{P}(Clusters) \sim Dirich(\alpha_1, ..., \alpha_k) \quad \text{where} \quad \alpha_1, ..., \alpha_k \text{are chosen from minimization of a utility function}$$
$$\mathbb{P}(existing\_cluster|clusters, entity) \sim Dirich$$

Algorithm Training

Training and Testing Datasets

We sampled 5 dates from a skewed right beta distribution ranging from 2010 to 2017. For each of these dates we scraped all the data from webhose.io and processed the text with Stanford's NLP algorithms and Microsoft Cognitive Services Text Analysis package. Each day's data was split up randomly into 5 folds for cross validation. Algorithm parameters were averaged across each of the optimal results from every day.

For each day, "fake" news sources were taken from a list of sources compiled by a Colombia university professor known to be fake. "Verified" news sources were taken from the top 100 most trusted U.S. news sources per a study by the Pew Research Center[1].

Results

Preliminary results indicate an accuracy of ~96%. Additional work is being done to optimize our clustering, cleaning our data, and choosing utility functions. More results coming soon.

Additional Information

Technologies Used

- Webhose.io
- Apache Tomcat
- Django
- Azure, Microsoft SQL Server, Microsoft Cognitive Services, Stanford NLP Libraries
- Javascript, HTML, Google Chrome Extension
- Developing Environments
    - JetBrains IntelliJ (Java)
    - JetBrains PyCharm (Python)
    - JetBrains DataGrip (SQL)
- Google Driver for file sharing, FB messenger for communication
- JetBrains YouTrack for bug tracking

[1] Trust and accuracy of American news organizations. Pew Research Center, 2016.

Primary Technical Contributions

Jay Khurana: SQL Server database loading, Database Schema design, Bayesian clustering algorithm
Christopher Chen: Django Server, Tomcat Server
Nathaniel Brennan: Database Schema design, Chrome Extension, all around technical help.
Harrison Xu: General algorithmic design and analyses
Julian Kelly: Company Logo and Chrome Extension UI Design

Created and Reviewed By

Jay Khurana
Chief Financial Officer
CrossCheck

Nathaniel Brennan
Chief Technology Officer
CrossCheck