Graham Dyer, CS 397, FA15 gdyer2@illinois.edu

This semester, Ismini and I continued our work on controversy-detection of news articles. Much progress was made in all areas, but most notably were efforts in building a dataset, improving the scoring algorithm, and implementing more precise retrieval of social content. In the beginning of the semester, the demo [1] was also improved.

Our work in building a dataset for training was the most apparent progress. The unannotated dataset consists of 1928 unique news articles including their metadata and comments, as well as relevant tweets (about 1 GB-worth). Retrieval of some tweets and all news articles is keyword-based. Thus, a single term is used to query and return a list of up to 10 news articles and 500 tweets. To get a diverse grouping of keywords (and therefore articles) that are controversial, neutral, and non-controversial, we used a list provided by Mejova et. al. in their 2014 paper "Controversy and Sentiment in Online News" [2]. We reduced their list slightly to 275 terms. These tweets and the URL of each article as well as attributes such as date, author, and the article's abstract are retained. We then scrape the article's webpage to obtained the full-text in addition to comments appearing on the page with their relevant metadata. Finally, a second twitter query is performed using the article's title without stopwords.

Non-tweet content is obtained from The New York Times, Reuters, and the Associated Press. Article webpage comments contain many metadata variables, all of which we save. In all twitter queries, we also save metadata related to the tweet itself (retweets, favorites, etc.) and that related to the author (location, followers, etc.). All of this content is stored in a persistent database and is optimized for training.

The largest obstacle in curating the unannotated dataset was speed. Given the considerable size of the data (and single-core nature of the server!), building an efficient scraper was key. However, minimizing the time spent on this aspect of the project (so we could start annotation and thus improve the scoring function) was equally important. Speed in the former case was improved by not running various analyses on the retrieved content. We do as little post-processing as possible: no scoring occurs, no linguistic or sentiment features are noted, and organizing into a schema-based database is avoided. Another consideration was not using a ranking function such as Okapi BM25 to filter tweets by their relevance to the articles. These procedures can be done later and, in the case of sentiment, more efficiently in a batch.

In addition to building the unannotated set, it's obvious that a method must be created through which humans can annotate (select controversial sentences) each article. This was done with a web interface [3] which attempts to verify quality. We take a few considerations into account:

- Tell annotators what to look for when determining whether a sentence is

[1]https://controversy.2pitau.org
[2]http://arxiv.org/abs/1409.8152
[3]https://controversy.2pitau.org/training

controversial

- Ensure a script is not interacting with the page by forcing a CAPTCHA before annotation is possible (browser cookie set so this only occurs once per annotator)
- Ensure articles that have already been annotated 3 times are not shown again
- Ensure one article is not annotated at the same time by multiple readers to avoid database collisions
- Prevent annotators from submitting the article with a reading speed faster than 800 words per minute (a very fast skimming pace)
- Prevent annotators from delaying another reader from viewing the same article (webpage is kept open longer than 300 word / min or closed before submitting)
- Increase readability by reducing interface clutter and adding paragraph separations
- Tokenize the article properly to go beyond classifying things like "Mr. " as a sentence
- Avoid very short articles

In addition to performing many of these checks within the annotator's browser, some of them are duplicated by the server to guarantee quality.

Once a fair amount of articles are annotated, we will run a classifier, such as a support-vector machine to determine those features (social, linguistic, etc.) that have the largest impact on a sentence's or article's controversy.

Various improvements were made in addition to those above. We've switched to SentiStrength, a sentiment analyzer designed specifically for tweets [4] for which we've obtained an academic license. The demo was also improved to work in more browsers. In the beginning of the semester, I implemented the linguistic features of the scoring function mentioned by the original full paper Ismini wrote. Since then, we've considered more diverse variables (like all the metadata we've collected for training) and are excited to see the affects of this machine-learning approach.

[4]For example, text features such as emojis are even evaluated