Web Claim Finder: A cluster-based approach for identifying disputed claims on the Web

Daniel N. Byler
School of Information
University of California, Berkeley
Berkeley, California
dbyler@gmail.com

Abstract

I present an automated approach to identifying the most widely disputed claims online. This method consists of performing web searches for trigger phrases (e.g., "falsely claimed that"), processing and clustering the results, and promoting tight clusters as candidates for "disputed claim" status. This approach results in fairly high precision clusters and may be adapted for large-scale data analysis.

1 Introduction

The Internet contains a vast array of web pages, blog posts, and articles which make factual claims about the world. Although all such claims reflect the point of view of their content creator, identifying points of dispute can be a challenging task. Dispute Finder is a tool that overlays a network of factual claims on top of the existing web, identifying to the user web claims that may be in dispute or otherwise of interest. This project attempts to augment the Dispute Finder project by identifying the most highly disputed claims on the Internet through automated means.

2 Related Work

This research is based on a number of well-established methodologies in natural language processing and machine learning, including input preparation and clustering. Adreopoulos et al (2009a) present an excellent analysis of clustering techniques in the wild, and their work led me to select a hierarchical clustering approach.

3 Data and features

Because the target data set for this project is "the most disputed claims online", I used the Yahoo! BOSS search API to assemble a corpus of extracts from Web sources. For this initial iteration, I used the query "falsely claimed that" to identify candidate web pages that might reference a disputed claim online. The first 750 English-language results for this query became the starting point for this corpus. For each search hit, the program attempts to scrape the text content of the source page to identify the context of the search phrase. For pages that could be successfully scraped, 250 characters of text beginning with "falsely claim" became the document's identifier; where this failed, the Yahoo! BOSS search abstract was used.

To avoid clustering identical results due to syndicated or plagiarized content, any duplicate documents were eliminated from the corpus. This reduced the corpus to 711 items.

Next, to select features for clustering, the algorithm eliminated the standard stop words included in the Python Natural Language Toolkit (NLTK), as well as a selection of words frequently used as page or section identifiers, e.g., "CNN", "factcheck.org", "politics", etc. It then uses the Porter stemmer to stem the words.

Finally, the algorithm performs term frequency/inverse document frequency (TF/IDF) analysis on each remaining word. Each word's TF/IDF score is then used as a feature in the vector-space model.

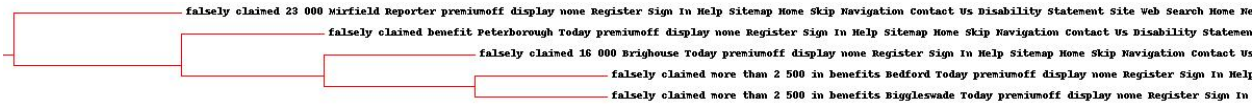

Figure 1. Promoted dendrogram represents a cluster of related claims

4 Models and Results

For the clustering procedure, I considered a number of algorithms and methods. The essential criteria for consideration were that the algorithm be fully unsupervised and that it provide adequate density measures with which to reject outliers from the clusters. K-means, often the first choice for clustering, fails to meet these criteria due to its requirement that the user preselect the number of clusters to identify. As the clusters in this research are unknown in advance, preselecting any criteria would be inappropriate. The HIERDENC algorithm (Andreopoulos 2009b) appears to offer an efficient, scalable and fully unsupervised approach to this problem; unfortunately, the implementation was unavailable at time of research. Ultimately I chose a DBSCAN clustering algorithm presented by Segaran (2007) due to its sufficiency with regard to the aforementioned criteria and its accessibility in Python.

The algorithm creates a hierarchical cluster by first considering each document in the corpus as an independent cluster. It then iteratively merges the closest clusters in vector space and repeats until each document falls under a single master hierarchy.

To identify target clusters, the algorithm next iterates through the tree and detaches clusters with a distance less than 0.75. These concentrated clusters are hypothesized to represent discrete claims. If successful, each document within a promoted cluster would represent approximately the same claim.

For the given corpus of 711 documents, the algorithm identified 31 clusters meeting the distance threshold. Of these, several included nested clusters; I disregarded these "sub-clusters", narrowing the scope to 23 non-overlapping clusters. A total of 51 documents total were cited in these 23 clusters.

Analysis of the results is here conducted according to standard precision/recall measures.

For a precision measure, I manually classified each cluster's documents according to whether they seemed to represent the primary claim of the other item(s) in the cluster. In cases where a cluster contained numerous documents, I compared each document to the cluster's centroid. Through this analysis, I identified 42 documents as successfully clustered, and 9 documents as unsuccessful, resulting in a precision rating of 42/51, or 82%.

In order to identify a recall measure, I manually searched for other documents within the corpus that, in my estimation, represent the central claim of each cluster. I selected the 13 clusters that were specific enough to search the corpus for additional matches with reasonable assurance of finding the relevant documents. Of these clusters, the algorithm had identified 30 total documents. Through my search, I identified 34 additional documents that represented analogous claims to the original 30 documents. This means the algorithm correctly identified 30 of 64 clusterable documents, giving it a recall rating of 47%.

5 Discussion

While implementing the clustering, I attempted two primary variants of this clustering approach. One variant was to use Segaran's code nearly verbatim. This approach did not incorporate any TF/IDF analysis, stemming, or other advanced feature extraction, but it produced a dendrogram consistent with others in literature: balanced, with expected forking occurring throughout the tree. Unfortunately, this approach had unsatisfactory results: although I did not quantitatively analyze the accuracy of this approach, even a perfunctory visual inspection revealed that most of the clusters were badly mismatched.

The second variant, whose results I described and analyzed above, involved a hand-coded method of creating the vector-space document matrix. This approach utilized stemming, stop word elimination, and TF/IDF, and produced satisfactory results; however, the dendrogram produced by this tree is oddly lopsided (c.f. Figure 2).

My overall methodology leaves ample opportunity for further improvements and research. Some of these potential improvements include:

- Data preparation: instead of collecting n characters surrounding the claim phrase, identify descrete sentences or paragraphs in the claim's vicinity. This could facilitate more accurate claim analysis.
- Feature extraction: use n-grams for features instead of single-word features. While narrowing the scope of cluster similarity, this would move the model from a “bag of words” approach to one that could potentially differentiate between opposing claims using similar vocabularies.
- Clustering algorithms: experiment with alternative clustering approaches. Improve efficiency to work at larger scale.
- Analysis: use Amazon Mechanical Turk for larger-scale results analysis.

Although much more research is called for, this paper demonstrates that even a bag-of-words-based clustering method can consolidate information from disparate sources into sets of related assertions. Although recall is low, precision is good enough to reasonably conclude that an analogous clustering approach may be a viable solution to the problem of identifying disputed claims on the Web.

Acknowledgments

My thanks to Barbara Rosario and Rob Ennals for their invaluable help on this project.

References

Bill Andreopoulos, Aijun An, Xiaogang Wang and Michael Schroeder. 2009. A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefings in Bioinformatics*, Vol. 10, No. 3: 297-314, 2009.

Bill Andreopoulos, Aijun An, Xiaogang Wang, Dirk Labudde. 2009. Efficient layered density-based clustering of categorical data. *Journal of Biomedical Informatics* 42: 365-376.

Toby Segaran. 2007. *Collective Intelligence.* O'Reilly Media.

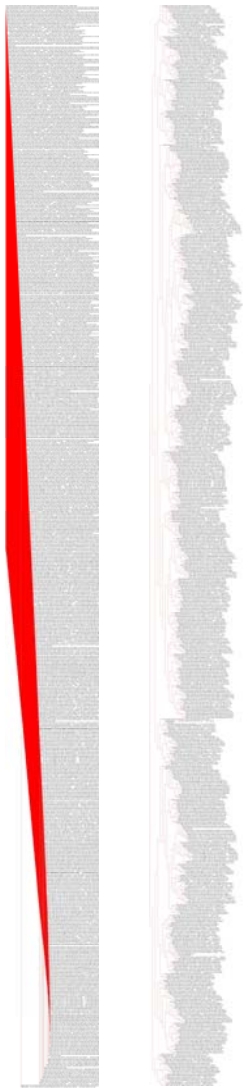

Figure 2: Two vector-space models result in very different dendrograms. At left, the dendogram resulting from my vector-space model; at right, the dendrogram from Segaran’s code.