Augmenting Rumor Detection Feature Sets with Source-Author Job Relevance

Alec Frey
Advisor - Dr. Barbosa
Department of Computer Science

Abstract

Deducing whether a tweet is a rumor or actual news is paramount in emergency situations. As soon as a Twitter user sends a tweet, their entire network becomes aware of what they said. Users have no clear way of knowing if a received tweet is truthful. Researchers have found key features within a tweet that are linked to credibility and are derived from both the textual content and author's profile. However, these features do not account for the specialty of the author. In crisis scenarios, few people have access to up-to-date information. By use of web scraping, we pull job information about authors of tweets from verified accounts. We then assign a similarity score based on their job compared to those that would be privy to current state of events. This similarity score increased average accuracy by 1% and bolstered precision, recall and F1 measure in all cases but one.

Introduction

Rapid proliferation of news is an integral part of our society. We all carry devices linking us to the rest of the world with the click of a button. In an emergency situation, we can rest assured that we will get information and help almost instantly. This is both a blessing and a curse. Misinformation can spread along with these bursts of news. Methods have been built across the realm of Natural Language Processing (NLP) to deduce when events are happening by using social media via interfaces know as Application Programming Interfaces(API). These interfaces allow researchers to capture data from all over the world during these key moments. Said researchers have had success in determining when an event triggers. They are even able to determine which tweets about the event were credible and which ones weren't. Unfortunately this analysis often happens after the event is over, sometimes hours after the event truly happened.

In the case of real-time emergency events such as a shooting, a fire, or an earthquake, there is not time to wait around and determine validity. It needs to happen in the moment. I propose augmenting current rumor detection feature sets with a factor for the author's specialty. As these situations happen incredibly fast, credible information will only be available to a select group of people closely related to the event, such as first responders and journalists. While the event unfolds, information provided by those in the know will be passed around from user to user and changed in minor ways. Without news stories to fact check tweets, we must instead look to the likelihood that an author is part of those responding to the event. To pursue this research I will focus on a single social network, Twitter.

Tools

Twitter Application Programming Interface (API)
Twitter provides researchers with the ability to harvest tweets using various methods within their API. These methods can be direct, by use of an ID specific to a tweet, or by listening to the stream of tweets being posted within a specific location. In both cases, twitter provides all the details related to a tweet, as well as information about it's author.

PHEME Dataset
Data set provided by Zubiaga et al [10] that contains annotated tweets. Each tweet is labeled as either a rumor or a non rumor. Within this dataset, there are tweets pertaining to five emergency events.

Beautiful Soup
Python library that can be used for parsing data from websites in a process known as web-scraping. It provides functions for easily navigating HTML tags to make extracting information quick and easy.

Methodology

Author Information Extraction
To ensure that we can find user profession information, we limited our dataset to only those tweets from verified accounts. As Twitter only verifies accounts of celebrities or those with massive followings, we deemed it a reasonable assumption that they would have a Wikipedia page. From these Wikipedia pages, we are able to pull information related to each author.

Using BeautifulSoup, we extracted the entirety of each Wikipedia page. We were then able to parse out the information we needed. Each page has a block containing summary information from the page. These blocks were our primary target for extraction. We used the text from the rest of the page to determine what words were most common using a Bag Of Words approach. The ten most common words were selected and added into our parsing result. As a final source, we used pattern matching on the first sentence, looking for common phrases such as “is a” and “was a” in order to deduce occupation.

Tweet Parsing
Each tweet was parsed and features related to the user were extracted for use in a Naïve Bayes Classifier. These features can be seen below and mirror those features used by Vijeev et al [9].

Feature	Description
Listed Count	Number of list a Twitter user is a part of
Average Number of Posts	Number of posts a user has made divided by their account age in years
Geo-Locations Enabled	Does the user have Geo-Location Services enabled?
Account Age	Ages in years of the user's account

Job Description Dataset
In order to build our similarity rating, we first needed to find text related to each job. We did this by using a corpus of job descriptions pulled from *dot-job-descriptions.careerplanner.com* [1]. Using Word2Vec's vector similarity tools, we compared job descriptions against a list of jobs we believed to have knowledge of up to date information. The highest rated jobs were then selected and used as vectors for comparison against an author's profession vector.

Word2Vec Similarity Ranking
Using the job descriptions and author information found above, we then built a similarity rating for each tweet. This rating was created by comparing our key words vectors and occupation vectors for each author against the related job vectors. For each author, we selected whichever comparison had yielded the highest similarity between the key words and occupation vectors. This rating was then added into our feature set listed above and ran through the classifier. The classifier was ran for 200 iterations and used a 75%/25% data split for training and testing. Results were averaged for each iteration. The same process was ran against the base feature set to get a baseline for results.

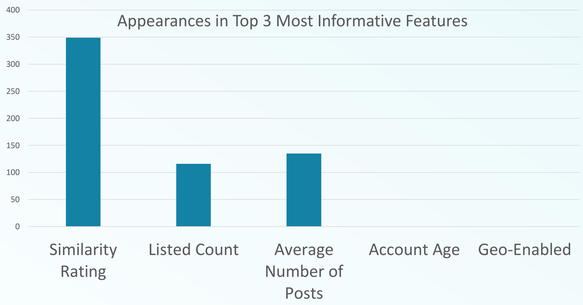
Results
Parsing information off of Wikipedia through the page's HTML proved to be difficult. As their website is not coded consistently, searching for items based on class names was not possible. This was coupled with issues finding proper pages for an individual when there were multiple pages for the same name.

Deducing job description vectors that were related to the jobs within our related list worked exceptionally well. Of the top 50 jobs we would select, there would only be one or two that had no relation to some form of first responder.

Similarity ranking was made difficult due to our restricted vocabulary for Word2Vec. This caused many of our job description vectors to be reduced greatly to contain only words in the vocab. This was exacerbated further by limiting our list of jobs to only those known by the vocabulary for comparison.

Despite these challenges, our similarity rating boosted the classifiers in every measure except one. We observed increase in the precision, recall and F1 measure for both our classifications. Our accuracy, only slightly out performed the base feature set with regards to average and maximum.

After each iteration, we retrieved the top three most informative features from the classifier. We found that our similarity ratings accounted for roughly 58% of these features.



Conclusions

Our approach to parsing Wikipedia information requires refinement. We encountered many issues with the html parsing. This could be remedied by using one of the Wikipedia APIs to pull a page directly. This would ensure we do not have to parse it ourselves.

Deducing related job descriptions worked quite well. The only modification to this section of the system is to build a method of dynamically determining what jobs are related to the event at hand. This in turn, would also require a process for classifying what event type a tweet is about at runtime.

Word2Vec similarity ranking performed adequately for our needs. In further research, it will be beneficial on all fronts to use a more verbose vocabulary. As we used the pruned vocabulary provided by NLTK (Natural Language Toolkit), many words from our job and author vectors had to be removed. Fixing this will bolster Word2Vec's ability to judge similarity.

All things considered, our slight success demonstrates that there can be improvement gains from using new author related features. By making the above corrections and adding in text based features, I anticipate our system can achieve greater results.

In a world of constant social interaction, it is imperative that we determine as quick as possible when a rumor is spreading online. I hope to have highlighted this need to my fellow students at MTSU and encourage them to press this issue.

References

[1] CareerPlanner.com,. Career planner.
[2] Castillo, C., Mendoza, M., and Poblete, B. Information credibility on twitter. Proceedings of the 20th International Conference on World Wide Web, pages 675–684, Jan
2011.
[3] Gupta, A. and Kumaraguru, P. Credibility ranking of tweets during high impact events.
Proceedings of the 1st Workshop on Privacy and Security in Online Social Media, Apr 2012.
[4] Gupta, A., Kumaraguru, P., Castillo, C., and Meier, P. Tweetcred: A real-time webbased system for assessing credibility of content on twitter. CoRR, abs/1405.5490, 2014.
[5] Jain, S., Sharma, V., and Kaushal, R. Towards automated real-time detection of misinformation on twitter. In 2016 International Conference on Advances in Computing,
Communications and Informatics (ICACCI), pages 2015–2020, Sep. 2016.
[6] Ross, J. and Thirunarayan, K. Features for ranking tweets based on credibility and newsworthiness. In 2016 International Conference on Collaboration Technologies and
Systems (CTS), pages 18–25, Oct 2016.
[7] Sato, K., Wang, J., and Cheng, Z. Credibility evaluation of twitter-based event detection
by a mixing analysis of heterogeneous data. IEEE Access, 7:1095–1106, 2019.
[8] Seifikar, M. and Farzi, S. A comprehensive study of online event tracking algorithms
in social networks. Journal of Information Science, 45(2):156–168, 2019.
[9] Vijeev, A., Mahapatra, A., Shyamkrishna, A., and Murthy, S. A hybrid approach to rumour detection in microblogging platforms. 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018.
[10] Zubiaga, A., Liakata, M., and Procter, R. Learning reporting dynamics during breaking
news for rumour detection in social media. CoRR, abs/1610.07363, 2016.