The Twitter Veracity Project

Raskesh Kumar, Phuong Viet Nguyen, Imani Palmer
University of Illinois at Urbana-Champaign
kumar19, pvnguye2, ipalmer2 @illinois.edu

***Abstract*—With the plethora of information streaming in via Twitter, it is crucial to determine the credibility of tweets in a real-time. We use machine learning and stream processing to build a prototype for a system that can ascribe credibility to an incoming Twitter stream. We start with exploring several machine learning techniques suitable for the features in Twitter data. We compare suitability of contemporary stream processing systems and use Spark's DStreams to build the prototype. We study the performance characteristics of the resulting prototype and establish its feasibility to provide a real-time credibility index.**

I. Introduction

Twitter is a source of constant communication and information. Recently, the stock-market flash crashed due to a fake tweet from a hacked twitter account of an otherwise credible news source[18] referenced in Figure 1. This incident and many others present a fundamental problem of trusting information on social media. There is a need for tools that enable users of these platforms to be able to attribute to some degree of trust in the received information. There have been several recent studies where machine learning has been used as a tool to describe the credibility of tweets [8], [9], [10], however these studies classify tweets offline and not in real-time.

This project entails offline processing, however, a bulk of the processing would be with a real-time stream. In addition, while some information (e.g., semantic content) can be leveraged to decide the credibility of a tweet, other information (e.g., topology of retweet graph) requires some time to be collected. Due to the massive size of the real-time problem[1], the solution needs to be robust and scalable. This paper explores the appropriate system design to do machine learning in the context of fusing offline and online processing. There is also an element of revisiting the classification of a tweets credibility as more information about the tweet is gathered (e.g., how many times it eventually got retweeted after being first tweeted and so on).

Fig. 1. Examples of Both Credible and Non-credible tweets

While it is impossible to computationally establish *The Credibility* of any information, we take a classification approach, where by tweets are deemed credible if they share features with labeled, verified examples. The classification indicates whether a tweet shares features with a true or false tweet and the confidence in the classification's result serves the purpose of the credibility index. We take into ac-

[1]According to a twitter blog[2], in 2013, an average day saw over 500 million tweets per day and the record for highest number of tweets pwer second (TPS) is 143,1991.

count various attributes of the given tweet, such as: semantic content, retweet structure and other tweet metadata. Through the usage of machine learning techniques, in combination with scale-out stream processing systems and offline data we develop a cascaded classification mechanism to attribute credibility to tweets of interest in real-time.

The rest of this paper is organized in the following way: Section II describes the related work. Section III describes our design principles and reasoning. Section IV describes our main experiments and results. Section V shows our conclusions and future work.

II. Related Work

There have been several recent studies in which machine learning techniques has been used as a tool to determine the credibility of tweets [20][7][21]. However, they only classify tweets in offline settings. Do to the best of our knowledge, a real-time credibility index using stream processing has not been developed.

Natural language processing (NLP) has also been used to determine mass emergency situations using Twitter[3]. Verma et. al demonstrated that using a classifier based on low-level linguistic features performs well in identifying tweets that help contribute to situational awareness. However, their techniques yielded high false positive and negative rates. Researchers at Yahoo conducted a experiment order the path of retweeted tweets based on news and retweeted tweets based on rumor. The results concluded that rumor tweets tend to be questioned at a higher rate than confirmed truths and that using text classifiers could be used to determine the credibility of tweets[7].

There has also been great strides in stream processing research. Storm[19] is a MapReduce-like system that provides a set of general primitives for doing real-time computation. Application logic such as ours is expressed using a Storm topology. A topology comprises of Spouts, which are data sources such as a Kafka/Kestrel queue. Spout gives rise to a stream which is a series a tuples and that is then fed into a Bolt which processes the tuples comprising the stream. Storm is simple, can be used with any programming language. Storm is also the mechanism of choice at Twitter. Bulk of Twitters analytics data pipeline is build using Storm[17].

III. System Design

The design for our main experiment is based off of a set of smaller initial experiments. These design decision are described in turn below:

A. System Architecture

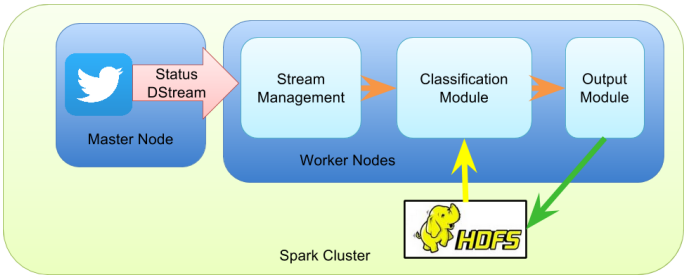

Fig. 2. System Architecture

The first part of design includes the usage of Spark Streams. Spark Streams[4] uses discretized streams (DStream), built on top of in-memory data structures called resilient distributed datasets RDDs[22]. DStreams are created from live incoming data (such as data from a socket, Kafka, etc.) or can be generated by transforming existing DStreams using parallel operators like map, reduce, and window. The decision to use Spark Streams instead of Storm arose from the need to both offline and online processing. Storm does not provide its own offline processing.The bulk of the processing in a real-time credibility assignment is done as the tweets are received. In addition, while some information (e.g., semantic content) can be leveraged to decide the credibility of a tweet, other information (e.g., topology of retweet graph) might require some time to be collected. Due to the massive size of the real-time problem, the solution approach needs to be robust and scalable. As demonstrated in Figure 2 our design streams in tweets through our master node with Spark Streams. DStreams are then sent to our three worker nodes for: stream management, machine learning classification, and output to offline handling on HDFS. The workers are also able to read in input from our offline model with HDFS.

This overall system architecture allows us to handle a dynamic mixture of online and offline processing.

B. Feature Identification

Tweets contain important meta-data. In order to determine which tweets are probablistically less credible, we determine tweet features that are interesting from the perspective of developing a credibility index. We conducted a small set of experiments in order to determine which features are necessary to assign a credibility index. These experiments employ a classification approach - to classify a tweet to be trusted, or untrusted, and use classification confidence score as the credibility index score. Twitter meta-data provides us with enough general information to use machine learning techniques to quickly classify tweets. These features are identified in Table I.

Overall, the features are divided into three categories: tweet-based features (i.e., features extracted from the content of the tweet), user-based features (i.e., features extracted from meta-data of user posting the tweet), and "expensive" features (i.e., features that are expensive to compute). The first two categories belong to the set of "cheap" features. The notions of "cheap" and "expensive" are based on whether the feature is computationally expensive to compute or not. This categorization will be used in our cascaded stream processsing model described in the following section. While some features in Table I have already been proposed in related works [12] [21], our contribution here is to categorize them in terms of its computational expensiveness and apply them in our proposed cascaded stream processing model.

C. Cascaded Classification

We observe that feature extraction for an arriving tweet can be divided in computation of less and more expensive features. We first reduce the velocity of arriving tweets by classifying them using low-cost features only. If a tweet is deemed classified with significant confidence, then we filter out the tweet and process the low-velocity stream through a more expensive features, hence avoiding to compute those features for all tweets.

Furthermore, the more expensive features also appear over a longer window of time. For example, the retweet graph evolves over time. Hence, we window the filtered stream such that while its velocity is reduced, it contains a larger interval, thus helping classification.

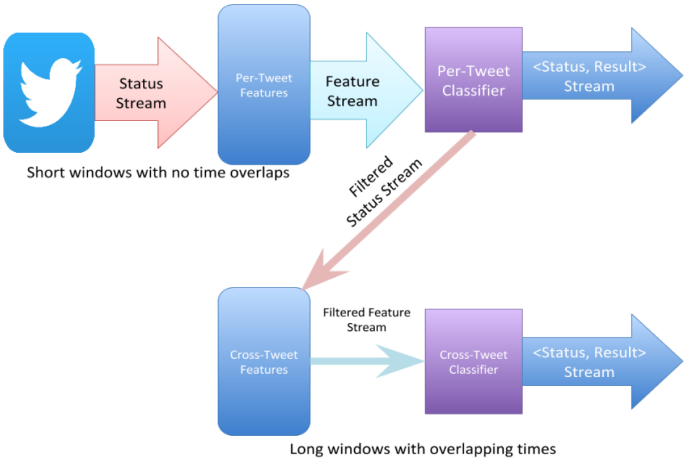

Fig. 3. Cascaded Classification

IV. Experiments

In this session, we do experiments to evaluate the effectiveness of our prototype in identifying the hoaxes from twitter stream, as well as the efficiency of the system when dealing with different levels of Twitter stream workload.

A. Experiment Settings

Our experiments is were conducted on two environments: i) on a local machine with 8-core CPU, 16 GB of RAM, and ii) on a Amazon Web Service (AWS) cluster that included one master, two workers, all are m3.large instances (i.e., with 2-core CPU and 7.5 GB of RAM each). This is to understand the effect of scaling out the stream processing system.

In terms of data, we collect multiple days of Twitter data on some trending topics in the US at the time of collection. Particularly, we focus on two popular topics: i) #MH370 - topic about the Malaysian missing flight, and ii) #forthoodshooting - topic about the shooting event at Fort Hood, Texas.

In terms of evaluation metrics, for classification task, we use traditional metrics, such as precision,

Group	Feature Name	Description
Tweet-Based Features	Is Re-tweeted	Whether the tweet is retweeted or not
	Re-tweet Count	If the tweet is a retweet, how many times it has been re-tweeted
	Favorite Count	How many times the tweet has been favorited
	Tweet Length	Length of the tweet, in characters
	Is Location Enabled	Whether the location of the tweet is enabled or not
	Hashtag Count	How many hashtags are used in the tweet
	Mention Count	How many mentions of other Twitter accounts are used in the tweet
	URL Count	How many URLs are used in the tweet
	Register-Positing Time Gap	The gap from the time user registers and the time user posts the tweets
Used-Based Features	Friend Count	How many friends does the user posting the tweet has
	Follower Count	How many followers does the user posting the tweet has
	Status Count	How many statuses does the user posting the tweet is verified or not
	Have description	Whether the user posting the tweet has account description or not
	Have URL	Whether the user posting the tweet has account URL or not
Expensive Features	Tweet Sentiment	What is the sentiment of the tweet (i.e., possitive or negative)
	Re-tweet graph statistics	Statistics of re-tweet graph (e.g., in-degree, ou-degree)

TABLE I. CLASSIFICATION FEATURES CATEGORIZED ON THEIR ATTRIBUTION TO INDIVIDUAL TWEETS/USERS

recall and ROC curve to evaluate the results, and use 5-fold cross validation to compute the values for each metric.

B. Classification Evaluations

1) Classification Techniques: We first experiment with a number of classification techniques to choose the one that is best suit for our task. There are four techniques that we are experimenting with, which are chosen from main classification categories and are the ones having wel-known performance on a wide range of problems [24]: i) Decision Tree - a tree-based method, ii) Logistic Regression, iii) Support Vector Machine (SVM) - both are function-based methods, and iv) Adaboost with Decision Tree as the base classifier - a meta-classification method.

We run each method againts three different topics: the #MH370, #forthoodshooting, and the combination of #MH370 and #forthoodshooting. The classification precision is shown in Figure 4. As shown in Figure 4, the meta classifier - Adaboost - achieves the best performance over all the topics, followed by the function-based techniques, and decision tree performs worst (except in the combined topic). In addition, it can be seen from Figure 4 that the result becomes worse on combined data of the two topics,

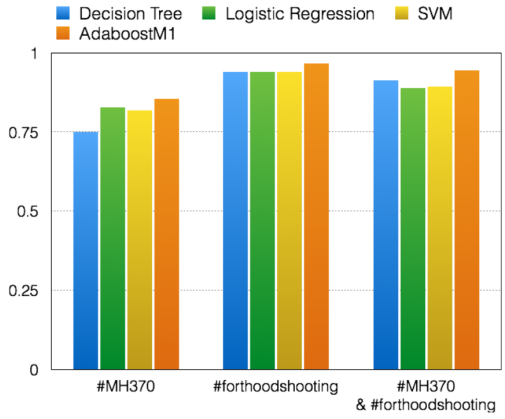

Fig. 4. Classification Precision Techniques Over Topics

compared with the results from separated datasets. This could be explained (and thus, verified) by our motivation from the beginning: since people could have different ways to discuss and spread the information on Twitter (e.g., think about political issues and disaster news), the significance of each feature and the classification model could be different for different topics. As a result, mixing the data from different topics cause the features negate the effects of each other, and negatively affect the classification results.

2) Cross-topic Classification: While the previsou experiment gives us information about the performance of techniques on individual topic, we would also want to see how each classfication technique can perform when it is trained by a topic and be applied to classify tweets from another topic. This is because, since there are so many topics available on Twitter, we could not be able to train the a new classifier model for every single topic, and thus, a classification technique that could perform consistently accross topics would be desirable.

In our second experiment, we trained the model for each techinques with one dataset of one topic (e.g., #MH370) and used that trained model to test with the dataset of another topic (e.g., #forthood-shooting). The result of this experiment is demonstrated in Figure 5. As we can see, the SVM technique does consistently better than other methods. This lead to our decision of using SVM as our main classification model.

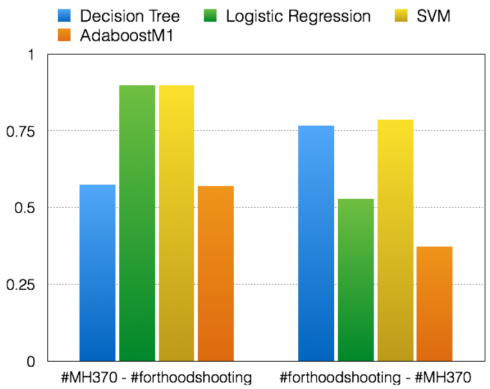

Fig. 5. Precision By Techniques For Cross-Topic Classification

3) Classification Result Overtime: In addition to the ability of cross-topic classification, we also want to see how the performance of one classification approach change over time without retraining the classifier. We sample the tweets (collected from #MH370 topic) from three separate days with 1 week gap, and train the classifier (we use Adaboost in the experiment, since it has the best performance for a single topic) using the data the first day. The trained model is then used to test with the data of the two days one and two week after that. It is interesting that (as you can see from Figure 6), the precision of the classifier does not change/decrease significantly (the green columns). However, the ROC (i.e., Receiver Operating Characteristic) area of the classification results decrease quite significantly overtime (i.e., the line in Figure 6). Since the ROC curve is drawn by plotting the fraction true positive rate and false positive rate, the ROC area result could tell us whether the classifier could achieve a good trade-off between the classification results of two classes (i.e., classifying as normal tweet or hoax). Specifically, the decreasing line of ROC area in 6 means that, overtime, although the precision of classifying normal tweet is quite consistent, the precision of classifying hoax decreases (which can be explained by the fact that we explore new hoaxes about #MH370 everyday). That is the reason why although the weighted precision is consistent, the ROC area curve still decreases.

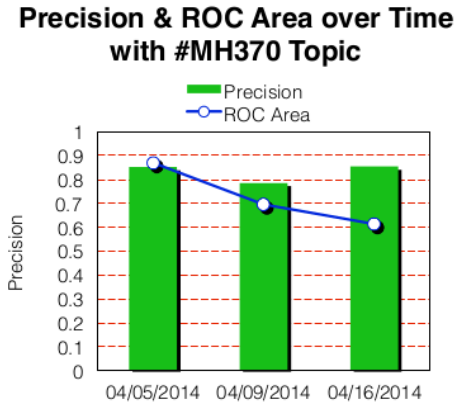

Fig. 6. Precision & ROC Area Over Time With MH370 Topic

4) Feature Importance: In this experiment, we measure the importance of feature in classification, i.e., how importance each feature in differentiating between normal tweet and hoax.

Since people can have different ways to discuss and spread information on Twitter (e.g., think about political issues vs. disaster news), the significance of each feature and the classification model could be different for different topics. Also, since the way people talk about topics on Twitter changes overtime, similar changes could happen with the importance of features. To measure the importance of each feauture, we calculate information gain (a

wel-known measurement) of each feature and rank the features in descending order. The results for our two experimenting topics are shown in Figure 7.

Most important features for #forthoodshooting	Most important features for #MH370
tweet_count	post_register_gap
follower_count	tweet_length
post_register_gap	mention_count
friend_count	is_verified
hashtag_count	follower_count
url_count	is_user_url
is_user_desc	is_user_desc
location_on	location_on
retweet	retweet

Fig. 7. Most Important Features

It is interesting to see the difference between the importance of features in each topic. Since #forthoodshooting is a newer event, and the cause of the event is unveiled quite fast right after the events, there are a fewer number of untrusted and false tweets on the topic, and thus, the imporant features for #forthoodshooting are those related to the popularity of user who posts the tweet (e.g., number of tweets, number of followers, friends, etc.). Whereas, #MH370 is an event that has been around for some time, and there are so many news source about it, and the stories are coming from all over the places. That is the reason why the important features for this topic are those related to the tweet itself, such as the lengthe of the tweet, the mentions appear in the tweet, and so on.

5) Evolution of Retweet Structure: As we described in the design of cascaded classification, in addition to the first round of classification using "cheap" features, all suspicious tweets are put into a second round of classification, where other more computationally expensive features (e.g., re-tweet topological graph statistics, sentiment of the tweet, etc.) are used to confirm whether the suspicious ones are really the untrusted tweets or not. In this part, we do a effectiveness study of the features in the second round of classification.

The "expensive" features, whenever a tweet is marked as "suspicious" by the first round of classification, place a tweet into monitoring system and more information is collected about the re-tweeting activity of that tweet. Particularly, all the re-tweet activities are represented as a directed graph structure. An example of this graph is displayed in Figure 8, the graph is designed to which each edge is a retweet, and each node is a tweet with some additional meta features from the first round, such as: follower count, favorite count, is-verified. This information is useful for studying the propagation pattern of a tweet, for example, to see if a tweet from a verified user is propagated in a different way from an unverified user, or how the favorite counts change with the propagation.

In addition, for each node in the re-tweet graph, also includes the sentiment of the tweet (i.e., whether the tweet is in positive or negative sense, given in a range from 0 to 4 with 0 as the most negative and 4 as the most positive). We dedicded to include this sentiment into re-tweet graph because the sentiment of the public should be consistent (i.e., either positive or negative) about the same topic. This aids in pinpointing the abnormal tweet through it having different sentiment from the common public about the same popular topic.

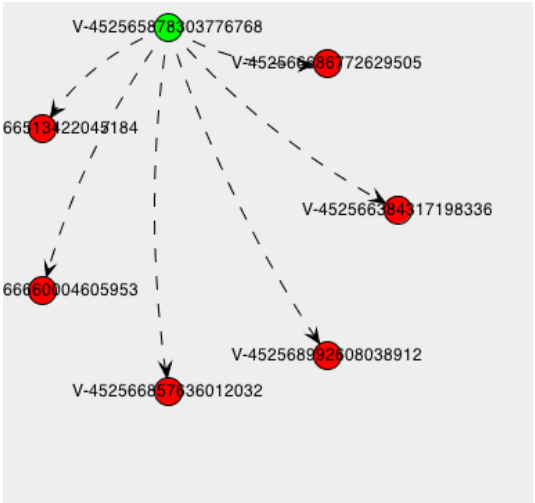

Fig. 8. ReTweet structure of a Tweet from a news agency about Flight #MH370

For retweet graph-based features, although during the data collection time, we did not observe any

significant retweet activity of untrusted tweets, we do observe a number of retweet activities for trusted ones and are able to identify some interesting patterns from the retweet graph. For example, the following is the re-tweet graph of a trusted tweet from a user about the missing flight MH370. This tweet receives 6 retweets during the data collection time (See: Figure 8). The colors in this case represent the sentiment of the tweet, green for neutral or positive, and red for negative. It is interesting that, since the user posting the original tweet is a news agency, it reports the news in a neutral language, whereas, the users who retweet the original tweet tend to express their negative thought about the fortune of the missing flight (See: Figure 9).

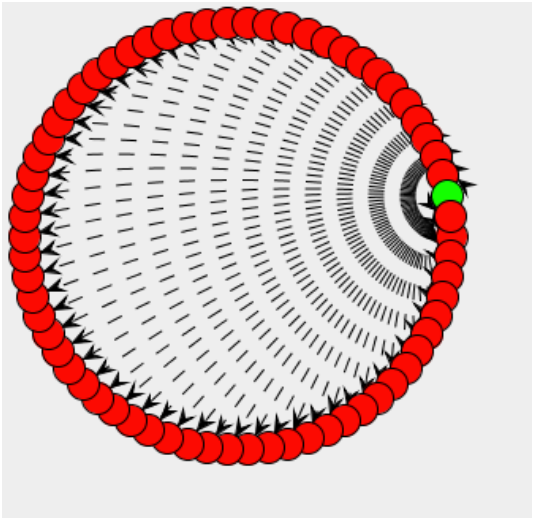

Fig. 9. ReTweet structure of a Tweet from a news agency about #FortHoodShooting

For the #forthoodshooting case, we plot retweeting graph of a trusted news, together with the is-verified status of user posting the tweet (green means user is verified, red means user is not verified). Since this is a new/breaking event, we observed much more re-tweeting activities (65 retweets). In this case, we again observe the star structure of retweet activity, and the tweet which is more likely to be retweeted tends to be from the verified user. In this case, the news agency reporting the event and posting the original tweet is the only verified user (green color).

These results suggest that retweeting structure, combined with meta information of nodes (e.g., is-verified status, sentiment of tweet) could give us further important information to understand the tweet better and be able to classify the trustworthiness of the tweet. These results also noted the most important features for each dataset. Even though these features are different they have a couple of similarities.

C. Performance Evaluation

We use Spark Streaming as the underlying fabric to build the classification application. The application itself contains a classifier which is loaded from a file and applied to arriving tweets. The training of the classifier was conducted offline using WEKA toolkit [23].

We use two workers and a single master node. In the data-center, all three are represented by threads running in Java processes on separate machines, while on a single node, these threads run in the same process. We allocate 4GB of RAM to each worker thread.

D. Synthetic Load

Twitter provides an Streaming API which is 1% of the entire stream. [1] The rate at which streaming API produces Tweets is uncontrollable and hence unsuitable for controlled evaluation of system. Hence we generated synthetic load to represent actual Twitter workloads on the same machine as the master thread. The synthetic load is consumed by the Spark Streaming application via loopback socket. The load constitutes of 1000 tweets captured from Twitter and replayed it in a loop to the system at specific velocities (tweets per second). We made the trace representative by mixing hoaxes, as well as arguably credible tweets and kept the format of the trace same as the output of Streaming API. We could control how long (number of seconds) the trace is replayed. Twitter sees an average of 5,700 (and growing) tweets-per-second, we evaluated the system up to 15,000 tweets-per-second load.

E. End-to-End Delay

Spark Streaming is a pseudo-stream process system in that it batches the arriving input and essentially performs batch operations on the input. We

chose the batch-interval to be 1000 milliseconds to have reasonably fast evaluation. The end-to-end delay for all tweets within the batch is the time it takes for the entire batch of tweets to be received and fully processed. For stable system operation this delay must remain under the batch-interval, however when the deadline is crossed, buffering on the receiving socket helps avoid loss of data.

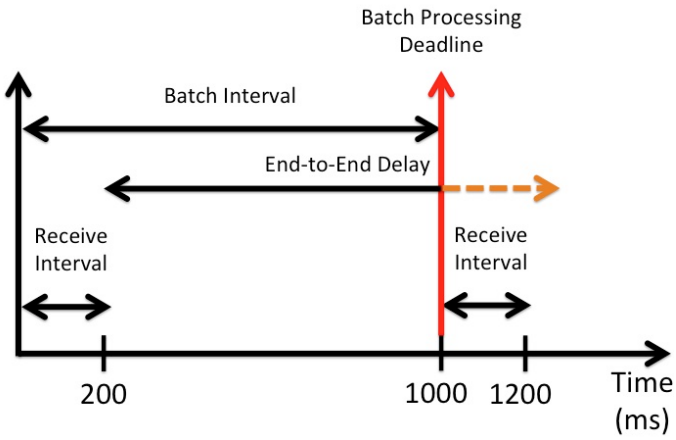

Fig. 10. Illustration of processing deadline

In our experiments (See Figure: 11), we observed our classification application to successfully process the tweets within the 1000 milliseconds deadline. However, we observe a near-linear growth in delay with increasing velocity. We were surprised by our finding that the results obtained in the local mode were better than what were observed in a three node cluster. We attribute this to the latency introduced by the communication of data and operation primitives from master to worker nodes.

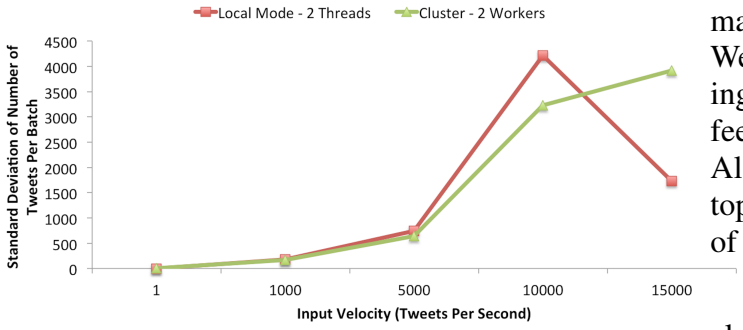

Fig. 11. End-to-end delay vs Velocity

F. Consumption of Load

To further study the dynamic behavior of the application, we measured number of individual records that constitute a given batch of input. This is a different number than the arriving load per interval because it depends on the processing capacity of the Spark Streaming application.

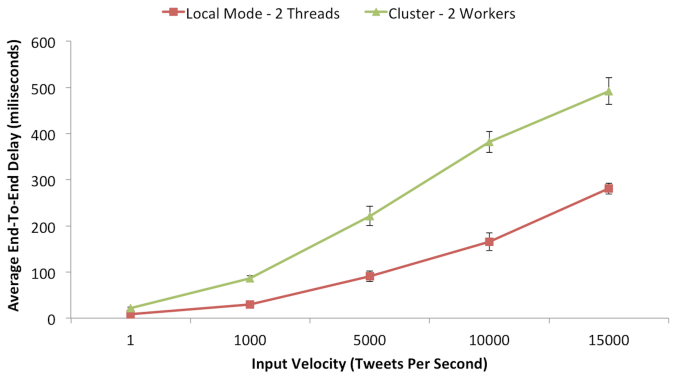

Fig. 12. Standard deviation of in Load Consumption

We observe that (See Figure: 12), while the application consumes input at a steady rate on low data-rates, there is large variation in consumption as the load increases even though we have employed a synthetic load which is homogeneous as input.

V. Conclusion and Future Work

We build a working prototype of a stream processing engine that provides a credibility index with the ability to scale. This prototype was tested on a cluster that was able to demonstrate feasible accuracy and scaling.

However, there are several areas where work may be done to improve classification precision. We developed an offline, trained classifier, but using crowd-sourcing and incorporation of human feedback, classifier can be trained online as well. Also, more testing of the classification on other topics/areas would needed to capture a wider variety of features.

Furthermore, testing must be done on a larger cluster while ingesting data from multiple sources to scale to shock-velocities of 150,000 as observed recently. [2]

REFERENCES

[1] Twitter Developers - Streaming API *https://dev.twitter.com/docs/streaming-apis/streams/* 2014.

[2] Twitter Blog: New Tweets per second record, and how! *https://blog.twitter.com/2013/new-tweets-per-second-record-and-how* 2013.

[3] Verma, S., Vieweg, S., Corvey, W., Palen, L., Martin, J., Palmer, M., Schram, Aaron, and Anderson, K. 2011. Natural Language Processing to the Rescue?: Extracting "Situational Awareness" Tweets During Mass Emergency. *Association for the Advancement of Artificial Intelligence.*

[4] Zaharia, M., Das, T., Haoyuan, L., Shenker, S., and Stoica, I. Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters. *USENIX HotCloud.*

[5] Zaharia, M., Chowdhury, M., Franklin, M., Shenker, S., and Stoica, I.2010. Spark: Cluster Computing with Working Sets. *USENIX HotCloud.*

[6] Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., and Stoica, I. 2012. Discretized Streams: A Fault-Tolerant Model for Scalable Stream Processing.

[7] Mendoza, M., Poblete, B., and Castillo, C. 2010. Twitter Under Crisis: Can We Trust What We RT? *Workshop on Social Media.* Washington, DC.

[8] Palen, L., Vieweg, S., Liu, S., and Hughes, A. 2009. Crisis in a Networked World: Features of Computer-Mediated Communication in the April 16, 2007, Virginia Tech Event. *Social Science Computer Review.*

[9] Vieweg, S., Hughes, A., Starbird, K., and Palen, L. 2010. Microblogging During Two Natural Hazards Events: What Twitter Contributes to Situational Awareness. *CHI.* Atlanta, GA.

[10] Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., and Widom, J. STREAM: The Stanford Data Stream Management System.

[11] Sankaranarayanan, J., Samet, H., Teitler, B,. Lieberman, M., and Sperling, J. 2009. TwitterStand: News in Tweets. *Association for Computing Machinery.* Seattle, WA.

[12] Vieweg, S. 2010. Microblogged Contributions to the Emergency Arena: Discovery, Interpretation and Implications. *CSCW.* Savannah, GA.

[13] Wang, H., Peh, L., Koukoumidis, E., Tao, S., and Chan, M. Meteor Shower: A Reliable Stream Processing System for Commodity Data Centers.

[14] McSherry, Frank., Isaacs, R., Isard, M., and Murray, D. 2014. Naiad: The Animating Spirit of Rivers and Streams. *SOSP.*

[15] Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M., Hellerstein, J., Hong, W., Krishnamurthy, S., Madden, S., Ramman, V., Reiss, F., and Shah, M. 2003. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World+. *Proceedings of the 2003 CIDR Conference.*

[16] Balazinska, M., Balakrishnan, H., Madden, S., and Stonebraker, M. 2005. Fault-Tolerance in the Borealis Distributed Stream Processing System. *SIGMOD.* Baltimore, MD.

[17] Marz, N. Apache Incubator Storm. *https://github.com/apache/incubator-storm.*

[18] Perlroth, N., and Shear, M. 2013. In Hacking, A.P. Twitter Feed Sends False Report of Explosions. *http://thecaucus.blogs.nytimes.com/2013/04/23/hacked-ap-twitter-feed-sends-erroneous-message-about-explosions-at-white-house/*

[19] Storm. *http://storm.incubator.apache.org/.*

[20] Castillo, C., Mendoza, M., and Poblete, B. Information Credibility on Twitter. *International World Wide Web Conference Committee.* Hyderbad, India.

[21] Morris, M., Counts, S., Roseway, A., Hoff, A., and Schwarz, J. 2012. Tweeting is Believing? Understanding Microblog Credibility Perceptions.*CSCW.* Seattle, Washington.

[22] Zaharia, M., Chowdhurry, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M., Shenkey, S., and Stoica, I. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. 2012.

[23] Hall, Mark and Frank, Eibe and Holmes, Geoffrey and Pfahringer, Bernhard and Reutemann, Peter and Witten, Ian H. The WEKA data mining software: an update, ACM SIGKDD explorations newsletter, 2009.

[24] Kotsiantis, Sotiris B. *Supervised machine learning: a review of classification techniques.* Informatica (03505596) 31.3 (2007).