Twitter Dedupe

- Sravan Bhamidipati (sb3400), PID8

Abstract

Twitter Dedupe is a low-latency clusterer that automatically divides any stream of tweets into different clusters based on their content. This is mainly aimed at helping users to handle the noise and manage the information overload on Twitter. If a user can see the timeline as different cluster, he/she can pay attention to tweets that are more interesting/important to them, or intentionally ignore tweets that are not interesting.

Introduction & Related Work

Twitter timelines are a constant flow of 140-character text messages called tweets. The popularity of the platform makes it very useful to get real-time news in times of emergencies and in general about all areas of a user's interest. On the downside, the heavy traffic which is often measured in Tweets Per Second, inevitably includes a lot of noise. Because the only current way to categorize tweets is through lists (of users), and noise is inevitably produced by users themselves -- noise here is not like spam, but can be any tweet that is not of the user's interest -- there is no easy way for a user to be on top of things, to filter out noise and to not miss tweets of special interest.

Twitter Dedupe tries to solve this problem by automatically dividing tweets into different clusters based on their content. Such clustering allows a user to get a quick overview of the entire timeline, and also to selectively filter out or highlight certain clusters.

Auto-classification is increasingly widespread. In addition to spam filtering, Gmail offers features like Priority Inbox to highlight important mail, and Smart Labels to label some mail as Bulk, Forums, and Notifications. Facebook uses Smart Lists to add friends to different circles. Twitter Dedupe is similar, but primarily challenging in a few different ways. Tweets, being brief and informal, are more difficult to make a context of. The number of data points in Twitter timelines (tweets) are far higher than in smart labels (emails) or smart lists (friends), contributing to issues of scalability as well as the number of clusters. The real-time nature of Twitter timelines implies the need for a low-latency clusterer to be able to divide tweets into clusters on the fly.

The project is an extension of a previous project that I worked on during last semester for the course "Cloud Computing: Concepts and Practice". The current project builds on top of its infrastructure (Google App Engine), but significantly differs from it in core intelligence. The previous work was limited to Twitter Search, had not used any NLP techniques, and clustered tweets based on text duplicity (identical tweets) and URLs (accurate but very slow). The new project aims to support user timelines as well (on authentication), and use NLP techniques to build a low-latency clusterer.

Data

There exist a few Twitter datasets in academia, though because of the Twitter Terms & Conditions these are released only with the status IDs and without complete texts. The status IDs can be used to retrieve the complete texts. I also wrote my own tools to collect Twitter data, and will add to that library. Though the availability of data itself is not difficult, crawling them will be harder because of Twitter's strict API rate-limiting. That said, using the Stream API will be suitable for the project's requirements.

NLP/ML Algorithms

At the tweet level, the first step would be to tokenize a tweet. Tokenization is needed to identify a subset of words that form the tweet which may contain information about the topic or content or context. One choice is to use a POS tagger, say, any of the taggers available as part of NLTK for Google App Engine. In case it turns out to be slower than expected I might maintain a small dictionary of common words, filter them out, and then look into the unigrams, bigrams and trigrams in the remaining words. For the time being I am considering these n-grams itself to be the topics because I am not sure whether there is enough data per tweet for topic modeling. (On the other hand using n-grams instead of topics might make the clusters too fine-grained and too many in number.)

At the timeline level, once the clusterer parses through N tweets and generates corresponding “topics”, I will use an unsupervised clustering algorithm (e.g. Canopy Clustering, K-Means, Expectation Maximization) to generate a few clusters. My current plan is to implement more than one algorithm and see which of them is most suitable for low latency without compromising on accuracy. If the unsupervised clustering gets too difficult or inaccurate, I also intend to try classification into common categories like Technology, Entertainment, Sports, Politics, etc. I do not yet know which one would be an easier (or better) approach, but my initial focus would be on clustering and not categorizing.

System Description

The project is going to be in the form of a website deployed on Google App Engine. The website will allow users to either login and see their own timeline, or without having to login search for keywords or see public timelines of users. Newly arriving tweets would automatically be pushed into one of the existing clusters. In case of people search, I intend to calculate some measure of the variety in the user’s tweets, which can be indicative of spam. (Spammers on Twitter tend to send a very small number of tweets to a very large number of people.) In the user’s timeline, I intend to have some metrics about each cluster, e.g. number of tweets, frequency of tweets, number of replies and retweets and favorites, etc. They are indicative of the “hotness” of a cluster (trending topics within a user timeline). It should also be possible to filter out all tweets and future tweets belonging to some clusters. To simplify the work to certain extend, I intend to only support tweets with English locale.

Conclusion

Twitter Dedupe aims at making a Twitter user’s life simpler in terms of managing the information overload. Using clustering, it allows the user to decide how to define noise and to filter it out, and also to highlight and pay attention to tweets of particular interest.