2016 4th International Conference on Future Internet of Things and Cloud Workshops

AffinityFinder: A System for Deriving Hidden Affinity Relationships on Twitter Utilizing Sentiment Analysis

Abdelmounaam Rezgui Daniel Fahey Ian Smith
Department of Computer Science & Engineering
New Mexico Tech
Socorro, NM 87801, USA
Email: {rezgui@cs, dfahey, ismi01}@nmt.edu

***Abstract*—Twitter is one of the largest and most popular social networking sites. While it has many interesting features, Twitter has no direct way to determine the relationship status between users. For example, unlike other social networks (e.g., Facebook), Twitter does not have any features for marking users as friends. In this paper, we present AffinityFinder, a system for automatically inferring potential friendship relationships (in terms of affinity) amongst Twitter users. The system collects and analyzes tweets to derive relationship scores that reflect affinity degrees amongst Twitter users. We implemented our tool using the TextBlob Python text processing library and the MongoDB database. Our evaluation shows that the system is able to derive potential friendship relationships with high accuracy. This system could provide useful data both to users and companies.**

I. Introduction

Social networks have become a worldwide phenomenon. They are increasingly connecting people from around the world. Their growth seems limited only by the number of people in the world. Their applications, however, are almost unlimited. For the provider of a social network, there are many ways to make profit from that social network. Often, this profit depends on the size of the network and on the features that the network offers to its users. An important feature is the ability to discover hidden affinity relationships amongst users. The network may then reveal these relationships to users, suggesting that they make these relationships explicit. This increases the network's connectivity and, in turn, the provider's profit.

A common problem in modern social networks is determining who may know who in the social network graph. Traditionally, friendship inference uses existing relationships (i.e., people you already know) to derive relevant suggestions about new friendships (i.e., people you might want to know). This type of analysis is relatively easy to conduct with different graph analysis metrics that use the already known friendship relationships. On Facebook, for example, users explicitly create these relationships. The system can then use these explicit relationships to compute recommendations for new friends. The problem, however, is more complex in social networks such as Twitter where there is almost no relationship information volunteered by users. Twitter is a social network different from what one may consider a traditional social network. It is less based on the relationships between users and focuses more on the consent that users develop and on creating a following of people who enjoy your Twitter content. Twitter provides a unidirectional relationship base where a user chooses who they would like to follow. Similarly, when someone follows you, you do not need to follow in return. This creates situations where the information is lacking in order to make informed inferences about who in the social network may know whom based on standard information provided by the users of the social network.

Users of Twitter do not provide traditional information about relationships held between them, their followers, or the users who follow them. This does not mean that the underlying relationships between the people who are following each other are not important. The networks provider must find ways to elicit these (hidden) relationships amongst users and then capitalize on them. There are naturally many ways that this can be done with the information that is provided to the service.

Significant research has been conducted on sentiment analysis in social networks. Most of the focus, however, has been on analyzing the sentiment of the writer of the tweet when he/she writes it. Examples of systems of this type include: Sentiment Viz, a system for tweet sentiment visualization [1], In [2], the authors present an approach to identify relationships between users and recommend who should a user follow. In [3], the authors introduce an approach for sentiment analysis on Twitter based on the idea of adding semantics as additional features into the training set for sentiment analysis. For example, one can add the semantic concept "Apple product" to each extracted entity iPhone from tweets. The authors also used the Naive Bayes classification in order to perform sentiment analysis.

In this paper, we present AffinityFinder, a system for automatically inferring potential hidden friendship relationships (in terms of affinity) amongst Twitter users. The system is based on the idea of using sentiment analysis to generate an affinity graph that captures likely affinity relationships amongst users based on the content of their tweets.

This paper is organized as follows. We first present our research goals. In Section 3, we present the design approach of the proposed AffinityFinder system. In Section 4, we present the architecture of the system and its implementation. In Section 5, we discuss our results. In Section 6, we conclude and discuss some future research directions.

978-1-5090-3946-3/16 $31.00 © 2016 IEEE
DOI 10.1109/W-FiCloud.2016.52

II. Research Goals

This prime goal of this research is to make smart inferences about relationships found in a social network based on non-traditional information. The information to be used is solely a collection of tweets that a person publishes and that contain mentions of other users. Using any tweet someone directs to another user, we conduct language analytics in order to classify the message under different classifications that will reveal information about the potential type of relationship those two individuals have with each other. This will produce an accurate representation of any relationship between two users based on their past conversations.

A system for affinity inference can be used in many applications. For example, one can use the affinity graph generated by the system to build a geoaffinity map that would give aggregate affinity information amongst residents of given locations (cities, counties) or between residents of pairs of locations (e.g., aggregate affinity between residents of New York, NY and San Francisco, CA).

Twitter has a function that allows a user to mention the names of others inside of his/her tweet. A tweet includes a twitter mention when the user specifies any user's Twitter handle that follows the @ symbol. Therefore, when a user user1 would like to direct a tweet to a user who goes by the name user5, they would compose the tweet as follows: "Hello @user5, how was your day?" Referencing people in this manner effectively has the same effect as using a person's name in a sentence. Mentions are therefore a mechanism allowing communication amongst Tweeter users.

A key idea in our work is to derive affinity scores between pairs of users based on mentions in tweets. This score measures the affinity degree between each pair of users. Note that many users can be specified in a tweet, i.e., a tweet can include many mentions. The result of this is that, when a user mentions multiple people in a tweet, the tweet will affect the relationship score between the sender and each person mentioned in the tweet.

III. Design Approach

The workflow of AffinityFinder consists of three phases: (i) data collection, (ii) Bayesian analysis of tweets, and (iii) building the affinity graph. We now elaborate on each of these phases:

A. Collection of Twitter Data

The process of data collection progresses in several rounds. In round 1, we start by selecting a random seed Twitter user. We then start harvesting tweets written by that seed user during a given time interval and filter out those tweets that do not include user mentions. As we collect theses tweets, we take note of any person that the seed user mentions in his/her tweets. If the seed user mentions a person in their tweet, that person is recorded in a list seed_users that contains the users to be used as seeds for the next round, i.e., users from whom we need to collect tweets during the next round. After the first round, we only have unidirectional tweets that can be analyzed to determine a relationship between the original seed user and any person with whom he/she communicated. This relationship analysis is only an analysis of what the seed user thinks about persons that they have talked to. In the second round, the same process is repeated for each user in the list seed_users. As existing elements are removed from the list, new elements are added to the list so that they are considered in the third round. This algorithm ensures that, after each round, the bi-directional relationship can be analyzed from both: (i) tweets that were directed to a specific user and (ii) the tweets that a user directed to the original person being followed.

The process continues until we decide that the collected data is sufficient to be used as a basis for an accurate evaluation of affinity scores. At that point, we start performing language analysis on the collected tweets in order to infer a relationship between pairs of users who may be communicating via Twitter mentions. In theory, the number of rounds can be as large as wanted. However, based on our tests, going beyond two rounds quickly creates a situation where the dataset grows at an exponential rate. We, therefore, limit our data collection process to two rounds.

In our experiments, we collected a dataset that consisted of approximately 12 thousand directed user tweets by approximately 600 users. Within these tweets, there were approximately 18 thousand relationships to analyze. This set of tweets was collected over the course of 25 minutes with substantial waiting in order to avoid hitting Twitter's rate limit.

B. Data Cleansing

The presence of "positive" terms may not necessarily signal positive sentiments from the sender to the receiver. For example, a sender may write: ".. I love this watch", in which case the sender's positive sentiment is actually directed towards the watch not the receiver. Also, the sender may write positive terms that signal positive sentiments towards other individuals than the receiver. In fact, when taken with other parts of the tweet, the seemingly positive sentiments may actually be completely negative towards the receiver. For example, a sender may write: "I admire Bob's courage but you and him are completely different."

C. Tweet Analysis

In order to perform analysis upon the collected tweets, we use the TextBlob Python library [4]. This library includes many useful features, including the ability to parse text into sentences and words. TextBlob also allows the use of a built-in Naive Bayes classifier to classify text depending on a set of training data.

Tweet analysis relies upon two metrics. The first metric is dependent on the classification of the tweet. Each tweet is run through a sentiment analyzer (word by word) using a Naive Bayes analyzer implementation. This will return a tuple of how positive and negative that tweet appeared to be. These two values will be summed such that the positive result will add to the sum and the negative result will subtract from the sum. Thus, a value of the overall "positivity" of the tweet is calculated. These individual tweet scores will then be linearly summed to create an overall score for the relationship between the two users.

The Bayes analyzer is given basic training data to calculate the positivity and negativity scores. This data consists of

basic phrases containing words that are seemingly positive or negative. Examples of positive words include “good”, “nice”, “please”, etc. Examples of negative words include “not”, “hate”, “horrible”, etc. It is important to note that the selection of these words is not based upon any studies or data. Thus, the training set selection still needs further improvements and revisions. The second metric is the total number of tweets between the two users. A higher value suggests a stronger relationship, while a low value suggests a weak relationship. This value will be used to scale the overall score computed by the first metric. Thus, a small number of overwhelmingly positive tweets will still register less than a long-lasting relationship that is slightly less positive. The general equation used for tweet analysis is:

$$F = N \times \sum_{i=1}^{N} (Pos_i - Neg_i) \quad (1)$$

where: F is the final friendship value and N is the total number of tweets sent from a specific user to another. Pos_i and Neg_i are (resp.) the positivity and negativity scores for the i^{th} tweet.

D. Building the Affinity Graph

The third step in AffinityFinder’s workflow is to build the affinity graph based on the affinity scores calculated during the analysis process (described in the previous section). To illustrate, we present in Figure 1 an example of what the affinity graph could look like. The figure (generated using Gephi [5]) is a detailed visualization of the relationships found between people. White nodes have a low outdegree while blue nodes have a large outdegree. The links that are green signify a good relationship. A white link is a neutral relationship. A red link corresponds to a negative sentiment relationship. This shows the potential of this work in producing real-time social network graphs of the relationships found on Twitter.

Fig. 1. Affinity Graph

IV. System Architecture and Implementation

Figure 2 shows an outline of AffinityFinders system architecture. We collect tweets using the Twitter API. Using a Twitter harvester that utilizes the Twitter REST API makes it possible to collect tweets from Twitter based on users’ Twitter handles. This API allows one to run a script to visit a users home timeline[1] and retrieve tweets from their page. We retrieve users’ tweets in a JSON format, filter out the irrelevant tweets, and then insert them into a MongoDB document database. Our choice of MongoDB is mainly for scalability concerns. With the scale at which tweet collection needs to occur, it is necessary to implement this system in a manner that can scale well to a big data framework. Indeed, MongoDB can be sharded if needed in order to distribute the tweets to different worker nodes. As descried earlier (Section III-A), when we increase the number of rounds in the data collection phase to more than 2, it will be necessary to use a fully distributed version of AffinityFinder capable of storing and processing gigabytes of Twitter data in a distributed fashion. A second reason for using MongDB is that it is natively able to store/query JSON documents which is convenient since native calls to the Twitter API also return JSON objects.

V. Discussion

Our experiments using AffinityFinder showed that it could collect, store, and process several millions of tweets with no major scalability issues. After filtering, the system was able to handle over 1800 relationships and had fully analyzed results from this dataset in under 20 minutes. Currently, the system runs on a single machine. Given the achieved performance, we are planning a fully distributed version that will be able to operate without limitations regarding the number of rounds. While effective on a small scale, the proposed system has some limitations. In particular, the Twitter API does not allow unlimited harvesting of tweets. There is a limit that is based on time for the harvesting of tweets. The limit is not easily reached but when expanding beyond two rounds of iterations of data collection, it becomes easily possible to reach the limit. Our next efforts will focus on developing strategies to overcome this limitation.

VI. Conclusion and Future Work

We presented AffinityFinder, a system for deriving hidden affinity relationships amongst Twitter users. Our major concern when developing the system was to design and implement a relationship analyzer scalable to large sizes. The system currently runs on a single node. However, it is designed to operate efficiently on big data scale. With this single-node implementation, the system is able to collect over 12,000 tweets over the course of less than 25 minutes. The tweets collected in the test dataset were analyzed for sentiment in less than 40 seconds. This time included the creation of a Bayes naive classifier. This shows the potential performance levels of the system when it becomes fully distributed.

We plan to continue our work on this system in several directions. First, we plan to develop a real-time version of the system where the MongDB database gets updated periodically

[1]A user’s home timeline displays a stream of tweets from accounts that user has chosen to follow on Twitter [6].

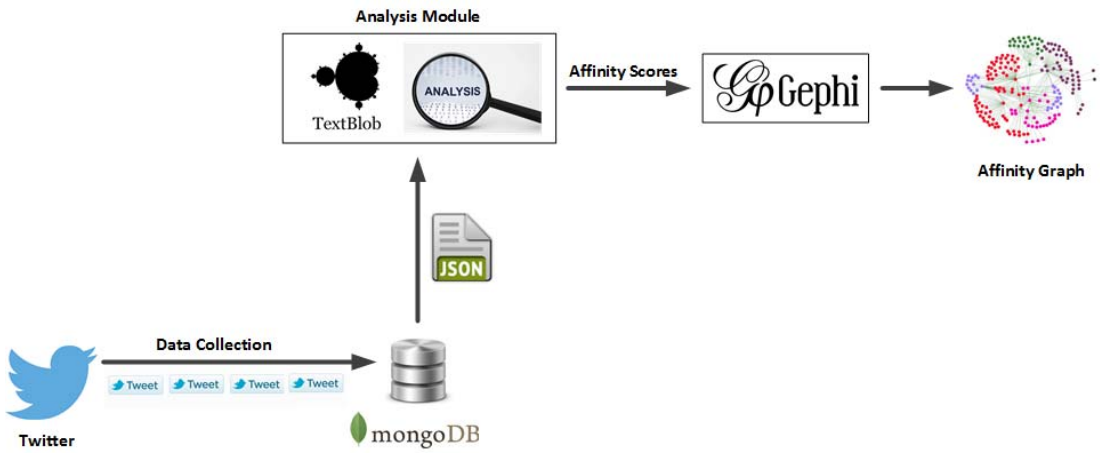

Fig. 2. Architecture of AffinityFinder

(automatically) with new tweets and the relationship scores are recalculated every few days or weeks. This will provide a more up-to-date score for the users. We also plan on developing a user interface that will enable users to quickly and automatically retrieve their scores. These scores could also be used to improve the user's experience by making tweets from users with high relationship scores more likely to be seen than other tweets. Another direction for future work is related to how the data is collected, stored, and analyzed. Data collection could be improved by pruning out certain "power users", such as celebrities and companies so that only personal accounts are considered. Performing more than two data collection rounds will cause the algorithm to go through tweets that it has already encountered, possibly wasting a significant amount of time. Also, storage can be improved by storing only the relevant data for each tweet instead of the entire JSON document as it is currently the case. Another direction is to use a more thought out training set that is specifically designed for the application at hand.

Finally, we plan to develop a fully distributed version of AffinityFinder. As of now, none of the algorithms takes advantage of distributed processing. However, we designed the system such that each component can easily be parallelized. For example, data collection can split up the work by groups of users so that each node collects tweets for one or a small group of users and adds those tweets to the database independently of the other nodes. Storage is already able to be distributed due to the nature of MongoDB. Also, the analysis phase can be distributed easily by simply dividing up the tweets between each node. This can be done intelligently by exploiting data locality or some other metric.

ACKNOWLEDGMENTS

The first author gratefully acknowledges travel support from the Institute for Complex Additive Systems Analysis (ICASA) at New Mexico Tech for presentation of this paper at the Third International Symposium on Social Networks Analysis, Management and Security (SNAMS - 2016).

REFERENCES

[1] "Visualizing twitter sentiment," https://www.csc.ncsu.edu/faculty/healey/tweet_viz, 2016.

[2] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh, "Wtf: The who to follow service at twitter," in *Proceedings of the 22Nd International Conference on World Wide Web*, ser. WWW '13. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2013, pp. 505–514. [Online]. Available: http://dl.acm.org/citation.cfm?id=2488388.2488433

[3] H. Saif, Y. He, and H. Alani, "Semantic sentiment analysis of twitter," in *The Semantic Web ISWC 2012*, ser. Lecture Notes in Computer Science, P. Cudr-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. Parreira, J. Hendler, G. Schreiber, A. Bernstein, and E. Blomqvist, Eds. Springer Berlin Heidelberg, 2012, vol. 7649, pp. 508–524. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35176-1_32

[4] Loria, Steven, "TextBlob: Simplified Text Processing," http://textblob.readthedocs.org/en/dev.

[5] "Gephi," http://gephi.github.io.

[6] Twitter, "Twitter's Help Center," https://support.twitter.com/articles/164083-what-s-a-twitter-timeline.