Adaptive Locality Sensitive Hashing for Recommending Twitter Followers

Siddharth Batra
Department of Computer Science
Stanford University
sidbatra@cs.stanford.edu

Abstract

Over the last couple of years, real-time information network Twitter has shot into the limelight amassing hundreds of millions of users. This magnitude of users and their tweets give insight into local and global trends but make it harder for users to find interesting people beyond the one size fit all celebrities. The aim of this work is to analyze a user's tweet history and find similar users the person might be interested in. The user tweets are treated as a single document and along with usage patterns are used to generate feature vectors. An adaptive approach to LSH families for the Euclidean distance is proposed for finding nearest neighbors in this high-dimensional space and the results are validated by treating the existing connections of a user as ground truth. As a secondary contribution the work also provides interesting statistics about Twitter usage patterns.

1 Introduction

Real-time information network Twitter [1] allows users to share updates with their followers and the world. This simple yet powerful mechanism allows millions of users to submit updates and things of interest enabling Twitter to leverage data mining techniques for discovering global and local trends.

The sad side of this beautiful ecosystem with a large asymmetric social graph is that most users feel clueless about whom to connect with. Twitter does provide a fixed directory of recommended users to follow in different categories but this one size fits all approach is a far cry from personalized recommendations.

The lack of recommendations of similar people creates a gap between content creators who want relevant users to consume their content and content consumers who want relevant content to consume. This also leads to bizarre attempts to find connections such as offering followers for sale on eBay [2].

Also note that the notion of similar does not imply that the recommended user will be exactly the same thereby making his/her content redundant but implies that in the high-dimensional space used to represent users some of the dimensions will be similar thereby opening the possibility of learning more about the similar dimensions and discovering interesting content in dissimilar dimensions.

With this motivation, the aim of this work is to present a framework for recommending similar users based on a person's tweeting history and usage patterns. A comparison to finding similar documents is proposed and the choice of the feature vectors is discussed in this light. An adaptive approach to locality sensitive hashing in Euclidean space is proposed for efficient and accurate

nearest neighbor search. This is used in conjunction with a supervised learning algorithm to enhance results. The work is validated by using the existing social graph of users on Twitter: note that this social graph is *not* leveraged during the adaptive LSH, a part of it is used as a training set for the supervised learning algorithm and the rest of is used as ground truth for the entire algorithm. As a secondary contribution a set of usage statistics are also presented about Twitter users that can be leveraged while designing feature vectors for various applications.

2 Approach

This section delineates the design of the various parts of the framework and motivates some of the design choices.

2.1 Users as Documents

> "Pelosi Signals Goal of Passing Health Measure Before Obama Trip http://bit.ly/9OxAs7"

The above text shows an example of a user tweet. Since by itself the text in a tweet doesn't offer enough information, all the user's tweets are combined together into a document to enable more robust features. The definition of a user document D_u is more formally illustrated in equation 1. Notice the use of summation rather than a union, this is done to preserve the word distribution.

$$D_u = \sum_{i=1}^{n_u} T_u^i \tag{1}$$

Given enough of a user's tweet history this formulation allows feature vectors to be built on top of standard text features used for documents.

2.2 Features

Before any algorithm for identifying nearest neighbors can be applied the raw data must be converted to a meaningful feature vector. The raw data available is a user's document D_u and date times for every tweet in the document.

2.2.1 Text features using D_u

The data is cleaned by removing special characters, stop words, URLs etc. since they aren't discriminative features and by applying stemming. Next, instead of trying to ascertain the top topics of a user's tweets which is essentially a hard filtering / prioritization of the available words, a soft estimate of $P(w|u)$ is used as shown in equation 2.

$$P(w|u) = TF_u^w = \frac{n_u^w}{\sum_i n_u^i} \tag{2}$$

Note the use of summation in the denominator rather a maximum as prescribed in the textbook. This form maintains a more probabilistic interpretation of the TF which is lost by using the max. The more robust TF.IDF metric was not used for logical reasons illustrated in the sections to come. Thus, equation 2 is used to assign a probability of occurrence to every word w depending on which user document it is found in. Although *not* mathematically equivalent, this can be thought of as the probability of a user u using the word w in a tweet.

References to other users and hash-tags are treated as text features. Although hash tags are meaningful but only 21% of users utilize them and inorder to not skew features towards a particular set of users using them as text features seems fair. For user references this approach can allow recommendations of users who may have a strong connection in common e.g. both are big fans of @ycombinator.

2.2.2 Date time features

The rough idea is that people who are similar will tend to tweet at similar times of the day and similar days of the week. Its a debatable feature to use but the formulation of the distance measure will compensate for this and give a high weight to the stronger text features. This translates to a seven dimensional feature using radial basis functions centered at 12:00PM on every day allowing for a softer measure of time periods rather than a hard discretization. These features are more formally illustrated in equation 3.

$$U(dt) = \bigcup_{i=1}^{7} \exp(-||c_i - dt||_2^2) \tag{3}$$

$$U(dt) \epsilon \mathbb{R}^7 \tag{4}$$

2.2.3 Feature vector

The probabilities of the words in D_u and the date time features together form the feature vector representation of the user. If we consider the unique number of words that can occur across all documents of all users then each user vector will be very high-dimensional. Of-course, this vector will also be very sparse since the probability mass is distributed in a select few words . Thus, for computational and storage efficiency the implementation only stores the words in D_u their probabilities computed using equation 2 and the date time features from equation 3.

2.3 Nearest Neighbor Search

This section presents the weighted Euclidean distance measure and the efficient adaptive LSH technique used for finding nearest neighbors or similar users in this very high-dimensional space of users that we have thus far represented sparsely.

2.3.1 Distance measure

Since all the values are between 0 and 1, we can use any distance measure in the Euclidean space for quantifying closeness between two user feature vectors. The L2 norm is chosen for its exponential nature of weighting i.e. similar users get very low scores and dissimilar users get a very high score. It is interesting to note that all the columns of the feature vectors may not be of equal importance, indeed we claimed previously that the day time features are not as important as the stronger text ones and this notion can be achieved by introducing weights as shown in equation 5.

$$dist(U_a, U_b) = \sum_{i=1}^{|U|} w_i^a ||U_a(i) - U_b(i)||_2^2 \tag{5}$$

$$dist(U_a, U_b) \neq dist(U_b, U_a) \tag{6}$$

$$\tag{7}$$

Note that the use of w in equation 5 is very similar to the notion of variance in the Mahalanobis distance [3]. So w can be thought of as a vector representing the variance of each dimension in this space.

2.3.2 Supervised learning to estimate w

Since this work explores personalized user recommendations, it seems natural to think of personalizing the weight vector w for each user. The idea is to see which features are more important for each user in finding people similar to them. The Twitter API [4] is used to find some of the user's connections amongst the 17,069,981 users available in the dataset. The feature vectors for the user's connections serve as positive examples and other users are randomly sampled to form a negative set.

The Kullback Leibler divergence [5] is then used as a criterion to rank the features using an independent evaluation for binary classification. The weights w then becomes the normalized value of the KL divergence. In the ideal computationally unbounded world, we would be free to compute a different weight for every feature but this requires an unrealistic amount of training data. Thus, the weights were used primarily to control the relative weighting of the text and date time features by giving all the text features the same weight. The results section presents more details, but irrespective of current API limitations it is an interesting idea to personalize w.

2.3.3 Adaptive LSH

The two main choices for finding nearest neighbors in an enormous high-dimensional space of 17,069,981 users and 476,553,560 tweets are clustering and hashing. Clustering users is perhaps a vague approximation for nearest neighbors since clusters can be large and ill-defined and they also can't support asymmetric distances. Plus, it is computationally quite expensive since it takes several passes over the whole set. The process of repeatedly testing for which clusters to merge make it inefficient even in a map-reduce setting.

Hashing techniques are a better choice in this light and the LSH family for Euclidean distances seems ideal. There are two main caveats for using this technique as is. First, for the hashing to work an index is needed for every feature. Given our efficient sparse representation this will mean having an enormous look-up table to look up indicies of the text features - this can get inefficient very quickly. Second, and more importantly it will be hard to introduce personalized weights for every user during the hashing stage - a notion of different bucket sizes for different features can perhaps be attempted but it still suffers from the former caveat.

With these caveats in mind an adaptive LSH technique for the Euclidean family is proposed. Adaptive implies that the hashing will be able to accommodate personalization by placing users in adaptive buckets using custom hash functions for each user.

The core idea is to create buckets on the fly by sampling the distribution of words of each user. Buckets comprise of singles,pairs and triplets of words. Equation 8 formalizes the probability of a user getting placed in an adaptive bucket.

$$P(B|u) = \frac{|B|}{Z} \prod_{b=1}^{|B|} P(w_b|u)P(w_b) \tag{8}$$

$$|B| \epsilon Unif(3) \tag{9}$$

$$B = \{w_1 ... w_{|B|}\} \tag{10}$$

The prior probability $P(w)$ can be computed using the inverse frequency metric and helps in down-weighting buckets formed by commonly occurring words. Equation 8 formalizes the probability $P(B|u)$ of a bucket B of size $|B|$ for a particular user u. Note that the $|B|$ in the numerator gives more weight to buckets with more words. Z is a normalization constant.

The bucket then serves as a key in the map-reduce framework and the idea is that two users fall in the same bucket only if their respective distributions had a high enough probability of a set of words that were uncommon enough - see equation 11. Note that this soft method of buckets enables serendipitous matching of similar users based on their interests in a subset of dimensions.

After the users have been adaptively hashed, all the users within a bucket are compared using their full feature vectors by the asymmetric distance measure in equation 5 and matches below a threshold are declared as recommended users.

$$P(u_1, u_2|B) = \frac{1}{Z} P(B|u_1)P(B|u_2) \tag{11}$$

Schema	Comment
raw text	raw file format with T,U,D on separate lines.
user,tweets,datetime	combined user tweets and usage patterns. 1MR.
user,D_u, $P(W\|u)$,datetime RBF,	feature vectors for every user. 1M.
bucket,Users,[D_u],[$P(W\|u)$],[datetime RBF],	all the users falling into a bucket. 1MR.
user,[users],total,[words],[distances],	final user recommendations. 1MR.

Table 1: Map reduce stages for computing user buckets.

User	Recommendations	Keywords
acct_jobsinhk	ir35accountant denaccounting accountancyage	accredited accountants manager limited company contract
aceball	ebookkeeper	tickets ballpark game financial credit
adriblue22	rachaelsmart	life work planned feel school alarm
aloredelam	silhouettefest	concert music festival video merci clip listening france
blackrain69	caribworldnews	headlines morning latest faces mocking caribbeans
cradlecatholic	carenetmilwauke	ultrasound weary posted feeling bible shalt good kill time lose reap

Table 2: Interesting user recommendations.

To sum up, the probabilistic approach to adaptive LSH enables an efficient yet accurate method of finding nearest neighbors in this high-dimensional space. It is interesting to note that this framework preserves the asymmetric notion of distance (i.e. user A might be recommended to user B but not vice versa) which is characteristic of Twitter but is hard to accomplish efficiently using LSH or clustering in their existing forms.

3 Experiments

The Twitter dataset provided by the course staff contains 476,553,560 tweets from 17,069,981 users. All the experiments were carried out on the Amazon Elastic MapReduce platform using HQL on top of Hadoop. The custom MapReduce scripts were written in Ruby.

The aim to run and validate all the theoretical experiments proposed for adaptive LSH on the entire available Twitter dataset required extremely efficient Ruby scripts and loads of High-CPU EC2 machines (20-50). The entire set of experiments on the entire dataset take 1.5 days to run.

Table 1 shows the schema of different tables that were created using HQL and populated with Ruby mappers-reducers. There were intermediate tables to hold the results from the map tasks but they are not as interesting.

Table 2 presents some interesting user recommendations. The keywords columns shows words that have a high joint distribution in both user documents. For the row with multiple recommendations it is not necessary that all the keywords will have a high joint distribution but in particular accounting case presented they simply happen to.

An unexpected application that emerged from this work was SPAM detection on Twitter. As soon as it became popular Twitter was bombarded with emarketers, porn sites etc. Experiments showed that at a very low distance threshold in the nearest neighbor search, a lot of SPAM accounts were exposed due to repeated tweets which had the same content over and over again. Table 3 presents these results.

Table 4 presents some interesting results about adaptive LSH and the overall dataset. The recommendation accuracy was computed by treating the existing connections of a user as ground truth and looking for % of recommended users that were existing connections. Its worth a mention that

Spam accounts	Keywords
surrealbrian2 surrealgirls	great exposure business girls
jasonbarczewski jasonbart	earn money online retweet plz
daveperryonline davep_etools davep_48	manchester offer portugal cris
blackberryking blackberrynews	blackberry storm curve review

Table 3: SPAM detection results.

Result	Statistic
Total users	17,069,981
Total tweets	476,553,560
Most active period	12AM-6AM
Most active day	Monday
% of users using #	21%
% of users using @	46%
Recommendations per user	32
Recommendation accuracy	$\geq 8.1\%$
Top 10 words	love good time will today going work great night check

Table 4: Overall results.

the Twitter API is extremely restrictive and allows only 150 calls per hour (sometimes less). This coupled with the fact that the dataset has user screen names where-as the API returns IDs meaning that an additional call is needed for every recommended user. This makes it nearly impossible to run validation experiments on a large set and hence only *this* particular result is validated on a smaller set (rest use the full set). Also note that is the lower bound of accuracy since there is no metric (apart from user testing) to judge whether a non-existent connection will be a good recommendation or not.

4 Conclusion

The real-time information sharing network has amassed millions of users and does a great job of discovering local and global trends. This paper attempts to address the problem of users not being able to find relevant people to follow apart from the one size fit all celebrities. A framework for implementing nearest neighbor search using asymmetric distances that mirror Twitter's asymmetric social graph is proposed by using adaptive LSH. A proposal to learn the weight vector w used in the Euclidean distance measure is also presented. The framework is validated on a 470 million tweet dataset and a lower bound on the recommendation accuracy is produced by treating the existing Twitter connections as ground truth. The recommendation accuracy is found to be $\geq 8.1\%$ and an average of 32 recommendations are provided per user. As a secondary contribution some interesting statistics about Twitter users are presented that can be leveraged while designing feature vectors for various applications.

References

[1] Twitter. Twitter About Us. *http://twitter.com/about*.

[2] E. Schonfeld. On eBay, Twitter Followers Are Worth Less Than A Penny Each. *http://techcrunch.com/2010/01/31/twitter-followers-ebay-penny/, 2010*.

[3] P. Mahalanobis. On the generalised distance in statistics. *Proceedings of the National Institute of Sciences of India 2 (1): 49.55, 1936*.

[4] Twitter. Twitter API. *http://apiwiki.twitter.com/*.

[5] Kullback, S. Leibler. On Information and Sufficiency". On Information and Sufficiency *Annals of Mathematical Statistics 22 (1): 79.86.*.