Discovering Similar Users on Twitter

Ashish Goel, Aneesh Sharma, Dong Wang, Zhijun Yin
Twitter, Inc.
@ashishgoel @aneeshs @dongwang218 @zjyin

ABSTRACT

This work studies the problem of discovering "similar" users at Twitter, where we define two users to be similar if they produce content similar to each other. The discovery of top similar accounts for each Twitter user has a variety of applications at Twitter including user recommendations and advertiser targeting. Although the discovery of similar results is a well studied problem in information retrieval, the particular problem at Twitter has three novel challenges. The first one is the heterogenous mix of signals that an effective technique could use: content analysis, social graph structure, user popularity, user interaction data etc. It is apriori unclear how could one blend all these different signals in an effective manner. Second, any technique needs to work effectively with very different kinds of users. This implies that the same framework needs to be applicable to users with millions of followers and users with very few or no followers. Finally, any proposed technique needs to be able to scale such that it can discover similar users for hundreds of millions of users, while keeping up with the highly dynamic nature of all the input signals. In this work, we share the machine learning based framework that we use to discover similar users at Twitter. We also evaluate the effectiveness of the framework on Twitter data.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithm

Keywords

Social Network Analysis

1. INTRODUCTION

Appearing in *Proceedings of the Workshop on Mining and Learning with Graphs (MLG-2013)*, Chicago, USA, 2013. Copyright 2013 by the author(s)/owner(s).

Twitter[1] is an online real-time social and information network that enables its users to send and read messages of up to 140 characters, known as Tweets. The service has rapidly gained worldwide popularity, with over 200 million active users as of 2012. Connections between the users are the core and lifeblood of a social media service, and the service itself can play a crucial role in helping users make such connections. One such mechanism is to discover top similar users for each user, which similarity be defined in a variety of ways. In this work, we focus attention on *production* similarity, namely the problem of discovering users who produce content similar to each other. We call this the "similar-to" problem.

Example 1 @KingJames (LeBron James) is a similar-to user account to @kobebryant (Kobe Bryant), because both LeBron James and Kobe Bryant are top basketball players at NBA.

Example 2 @katyperry (Katy Perry) is a similar-to user account to @ladygaga (Lady Gaga), because both Katy Perry and Lady Gaga are famous singers and songwriters.

The discovery of top similar accounts for each Twitter user has a variety of applications at Twitter. First, similar accounts can help in providing link recommendations which can drive the discovery of connections on Twitter. For example, if a user follows Kobe Bryant at Twitter, the user might also be interested in LeBron James. In fact, this benefits the recommended user as well, since users with more follows are more likely to be engaged with Twitter. Second, similar user accounts are also useful for accurate advertisement targeting since Nike might be interested in showing its advertisements to followers of accounts similar to itself, such as those of Adidas. Third, the discovered similar users could help enhance user experience by delivering more relevant content to users by helping target content recommendations better. Finally, since the social graph itself can be enriched by similar-to semantics, it can enhance the precision of various analytic tasks such as community detection.

Although the similarity problem is a well-studied one in the information retrieval literature, we note that there are several novel aspects to the particular setting of Twitter. First, Twitter has hundreds of millions of users, and any strategy has to scale to that size. Apriori, it is unclear if there is a scalable solution to the similar-to problem. To make it even more challenging, the social graph is very dynamic as connections are constantly being added or deleted.

[1] http://twitter.com

As a result, the similarity measure needs to be updated frequently enough to keep up with the change in a scalable way. Second, there is a multitude of signals that one could possibly use for determining whether two users are similar: the graph of connections, interaction information, content that users Tweet, etc. Taking all these pieces of information into account in a consistent way is a challenging task, especially at a large scale. Third, given the large number of users, it is expected that they are not homogeneous, and a good similar-to discovery system would take that into account. As an example of users having different characteristics, Lady Gaga (@ladygaga) has more than 30 million followers, while the most popular author on this work has less than 3000, which is a difference of several orders of magnitude. Therefore, any proposed algorithm needs to be able to address this heterogeneity in a consistent way.

In this paper we share the framework we use to solve the similar-to problem at Twitter. It is a machine learning based system on Hadoop, which is highly scalable and can discover similar accounts for hundreds of millions of users on a daily basis. The framework consolidates the information from social graph structure, tweets and logs by using a logistic regression model, and segments users into different groups for independent training. We also show the effectiveness of our framework via applications to a number of products in Twitter. It is important to note that we measure the effectiveness of our algorithms by ceding to the wisdom of crowds, and consider two users as similar if Twitter users who consume their content think that they are similar.

The paper is organized as follows. We discuss the related work and the background of similar-to product at Twitter in Section 2. In Section 3, we introduce the similar-to framework at Twitter in detail. In Section 4, we demonstrate the effectiveness of our framework via both real product applications and offline analysis. We conclude the paper in Section 5.

2. BACKGROUND

In this section, we summarize the related work to similar-to problem including user similarity metrics and user recommendation in information networks. And then we discuss about the similar-to problem at Twitter.

2.1 Related Work

2.1.1 User Similarity

Similarity measures have been extensively studied in the field of information retrieval and networks, and similarities between two entities are used for community detection, similarity search and recommendations. Most of the similarity measures proposed for networks are based on graph structure. In particular, SimRank [18] is a general similarity measure that is based on a simple and intuitive graph-theoretic model. The intuition behind the SimRank algorithm is that two objects are similar if they are related to similar objects. P-Rank (Penetrating Rank) [37] enriches SimRank by jointly encoding both in- and out-link relationships into structural similarity computation, which penetrates the structural similarity computation beyond neighborhood of vertices to the entire network. In [10], a scalable link-based Monte Carlo similarity search algorithm is proposed to approximate SimRank scores with a near linear external memory method and parallelization techniques sufficient for large scale computation. PathSim [30] is a meta path-based similarity, where a meta path is a path consisting of a sequence of relations defined between different object types. Meta paths give users flexibility to choose different meta paths and their combinations based on their applications. In [11], the quantities such as the average commute time and the pseudoinverse of the Laplacian matrix of the graph are used to compute the similarities between any pair of nodes, having the nice property of increasing when the number of paths connecting those nodes increases and when the "length" of paths decreases. Graph-based similarity measures are often coupled with clustering algorithms. LinkClus [33] takes advantage of the power law distribution of links, and develops a hierarchical structure called SimTree to represent similarities in multi-granularity manner. It avoids the high cost of computing and storing pairwise similarities but still thoroughly explore relationships among objects. RankClus [31] is a novel clustering framework integrating clustering with ranking in network, in which conditional ranking is used as new measures for clustering. We note that all of the above studies solely work on the network structure. Some recent work does use both graph and attribute properties. SA-Cluster [38, 6] uses a unified neighborhood random walk distance measure based on both structural and attribute similarities for clustering. Besides user similarity computation, some studies learn the effects of user similarity in social media. In [1], Anderson et al. studied how similarity in the characteristics of two users can affect the evaluation that one user provides of another.

2.1.2 User Recommendation

The similar-to problem is closely related to the user recommendation or the link prediction problem in networks, i.e. the problem of predicting the existence of a link between two entities, based on attributes of the objects and other observed links [16, 13, 14]. In [25, 26], measures of the "proximity" of nodes in a network are extracted from network topology alone for link predication. To conquer the limitation of social graph, a unified framework for link recommendation is proposed to combine both user attributes and graph structure in [34, 35]. Besides positive link prediction, in [24] both positive and negative links are predicted by casting the problem as a sign prediction problem, which connects to social-psychology theories of balance and status. All these studies of user recommendation are based on unsupervised approaches. In our similar-to framework, we use a supervised approach that learns from the historic data. Lots of the studies in user recommendation/link prediction are supervised learning based, including relational Markov network [32], probabilistic evolution model [19], supervised random walks [3], spectral graph transformations [22], etc. In [20], a semi-supervised link prediction method is proposed by applying the label propagation technique to link prediction.

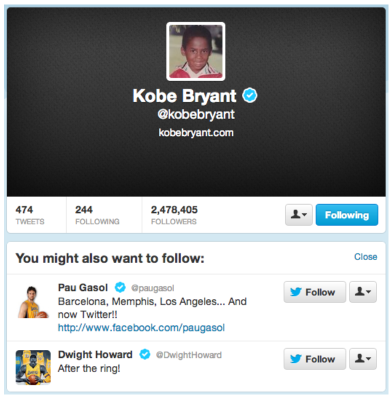
The link prediction problem is also closely related to collaborative filtering. In [21], a neighborhood model is proposed for collaborative filtering by minimizing a global cost function to exploit both explicit and implicit feedback by the users. A key difference between our similar-to framework and these studies is that we use a logistic regression model to consolidate multiple signals in a scalable manner. A regression model based approach is also used in [7] for movie recommendation, but the feature set is irrelevant to similar-to problem at Twitter and it is unclear if their implementation is scalable.

Finally, we also note that Twitter's user recommendation service (Who To Follow) was introduced in [15], which is responsible for recommending potential follows for each user. Our similar-to framework solves another fundamental problem, namely that of discovering top similar users for each user. As in Example 1, @KingJames(LeBron James) is a top similar-to account for @kobebryant(Kobe Bryant) based on our similar-to framework, but @KingJames might not be a good suggestion for @kobebryant to follow. To the best of our knowledge, our paper is the first one to address the specific similar-to problem at the scale of hundreds of millions of users.

2.2 Similar-to at Twitter

Our similar-to framework powers up many products at Twitter, including "You might also want to follow" and "similar to" on both Web and mobile platforms, similar-to suggestion email notification, etc. Besides these straightforward applications, the similar-to framework also helps improve the relevance of other Twitter services. For example, the similar-to computation is able to improve the quality of user recommendations via suggesting users similar to ones they already follow.

2.2.1 *You might also want to follow*



Barcelona, Memphis, Los Angeles... And now Twitter!!
http://www.facebook.com/paugasol

After the ring!

Figure 1: You might also want to follow module on twitter.com for @kobebryant (Kobe Bryant).

If a user visits someone's Twitter profile page and follows that account, similar users will pop up as suggestions under the text "You might also want to follow". Such an example is shown in Figure 1, where a user visits the profile page of @kobebryant, and on following that account, similar user accounts such as @paugasol (Pau Gasol) and @DwightHoward (Dwight Howard) are shown as suggestions to follow.

2.2.2 *Similar-to module on mobile*

If a user views someone's Twitter profile on mobile and scrolls down, they will find a similar-to module that shows user accounts similar to the profile account. For instance, in Figure 2, such an example view is shown for @ladygaga's profile page on the Twitter iPhone App, where @taylorswift13 (Taylor Swift), @britneyspears (Britney Spears) and @JLo (Jennifer Lopez) are shown as the result of "Similar to Lady Gaga".

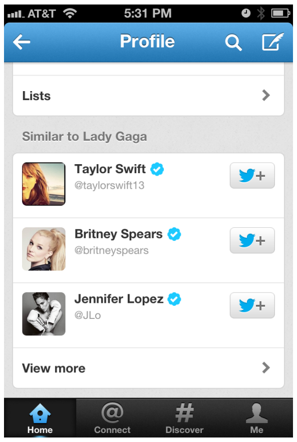
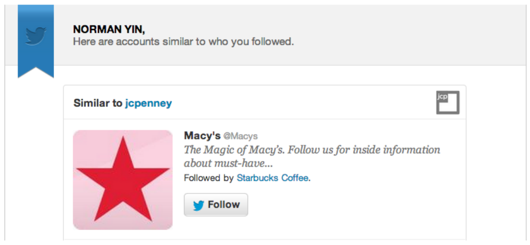


Figure 2: Similar to module on Twitter Iphone App for @ladygaga (Lady Gaga).

2.2.3 *Similar-to Email Notification*


NORMAN YIN,
Here are accounts similar to who you followed.

The Magic of Macy's. Follow us for inside information about must-have...
Followed by Starbucks Coffee.

Figure 3: Email notification of for similar to @jcpenney (JCPenney).

If a user follows some account, Twitter also sends the user an email with suggestions of similar accounts to follow. For example, if a user follows @jcpenney (JCPenney) recently, Twitter might send an email with accounts similar to @jcpenney such as @Macys (Macy's) as shown in Figure 3.

3. SIMILAR-TO FRAMEWORK

In this section, we introduce the similar-to framework at Twitter. Our framework is mainly implemented in Pig [28] on Hadoop. Pig is a high-level data flow system between SQL and Map-Reduce [12]. Pig programs are compiled into sequences of Map-Reduce jobs, and executed in the Hadoop Map-Reduce environment. The design makes our framework highly scalable for the use base of the size of hundreds of millions. As shown in Figure 4, we have a "pipeline" of jobs, which consists of three components: candidate generation, model learning and regression, which run on Hadoop on daily basis. In the candidate generation step, for each user account, we generate a candidate set of potentially similar users based on social graph. In the model learning step, we train a logistic regression model based on historic log data. In the regression or the classification step, we take the generated similar candidate set for each user, apply the trained model and obtain top similar users. The candidate generation process is described in more detail in Section 3.1 and the model learning is described in Section 3.2.

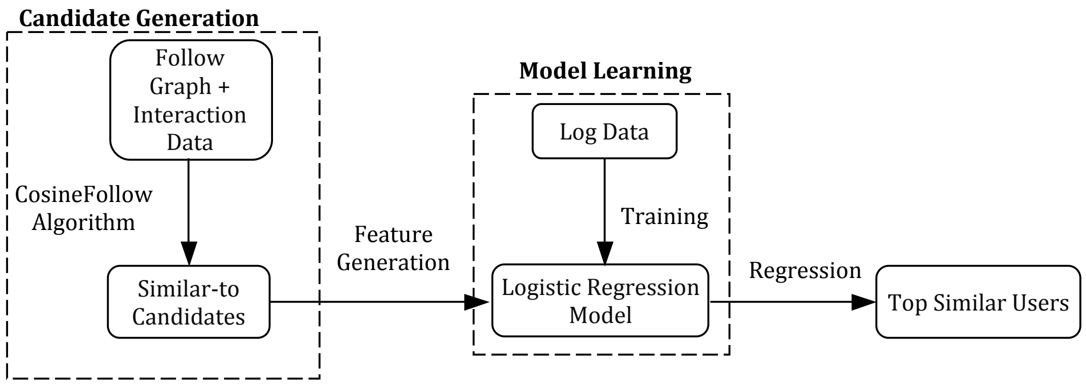


Figure 4: Twitter Similar-to Framework.

3.1 Candidate Generation

As stated above, the similar-to candidates for a user are generated based on whether they have a similar set of followers. This is achieved by computing cosine similarity on the Twitter graph of follow edges. The graph of follow edges is Twitter's social graph, where an edge represents a following relationship. Then, we can define two users to be similar using the cosine similarity of their sets of followers. We also note that the graph being used here does not need to be only the following relationship. For instance, if a user A retweets a user B frequently, then this information can be added to the graph of connections. In particular, we can define a union of graphs and represent it using a weighted graph, where the weight of an edge represents the strength of relationship. A logistic regression model is learnt so that the weights equals the probability of future interactions between users. This model uses features on the edge such as past interactions, common friends, interest match etc. It also includes user level features such as overall activity, global reputation etc. We mention that to emphasize that this weighted graph has already captured a variety of signals in addition to the graph of connections. And of course, the cosine similarity notion is easily extended to take the weights into account.

3.1.1 Cosine Similarity on Hadoop

This section talks about how to compute top-k cosine similar users from a given graph on Hadoop. To understand the scale of this computation, consider the Twitter connections graph, which has hundreds of millions of nodes, and tens of billions of edges. A naive all-pairs similarity computation on such a graph is completely infeasible, so one has to think about some optimizations. Furthermore, it is not at all obvious how to apply locality-sensitive hashing [17] based, or HyperLogLog-based [8] techniques since a primary source of complexity for the problem is that a lot of different pairs might be similar and just materializing those pairs on Hadoop results in a shuffle size of hundreds of terabytes. Also, simple optimizations like excluding high degree accounts from the computation is not feasible as these accounts may significantly affect the graph's connectivity and hence the quality of results. The work closest to our approach is the DISCO algorithm [36], which our work predates, but that work also focuses on exactly the shuffle size which is the primary source of complexity in the Hadoop setting.

In this work, we've proposed a two-step random sampling approach, and to compute top-k most similar users, we run the computation below. Recall that we have a directed weighted graph $G(V, E)$ with weight of an edge $e \in E$ given by $w(e)$. Now the computation is as follows:

1. For each node $u \in V$, compute a sample of u's followings, i.e. sample edges with replacement from all nodes $v \in V$ such that $(u, v) \in E$. Call this sample $S(u)$. This is weighted sampling, and the weight used for sampling v is set to $\frac{w(u,v)}{\sqrt{\text{InDegree}(v)}}$. Note that this weight is part of the denominator of the cosine similarity formula.

2. Propagate a new sample $S(u)$ to each of its followings, i.e. to all nodes w such that $(u, w) \in E$.[2]

3. After receiving the samples from all of its followers, each user $w \in V$ then aggregates the samples $S(u)$ from all u. First, we multiply the score of each sample $x \in S(u)$ by $\frac{w(u,w)}{\sqrt{\text{InDegree}(w)}}$, then and sum up the total score for a given user x across all the samples. Then, the top-k scoring users are considered the most similar users to w. Note that the aggregated counts are equal to the cosine similarity formula in expectation.

We also apply various heuristics to scale the implementation and improve the quality of the approximation. For example, for a popular user x with millions of followers, we sample its followers, and only aggregate subsamples from a selected subset of followers. Furthermore, when x aggregates subsamples from one of its followers y, the number of subsamples used is determined by the popularity of y, where the popularity is measured by pagerank on the graph.

3.2 Model Learning

Note that although we did use multiple signals in generating the initial set of candidates, they had to be compressed into a single weight value on a graph edge. To provide more resolution for consolidating multiple signals, we also

[2]In practice, generating a new sample for users who follow a lot of users is very expensive, so instead we sub-sample from a large initial sample as an optimization.

train a logistic regression model based on log data of "You might also want to follow" as shown in Section 2.2.1. We use the features from graph-based algorithms, suggestion's popularity, information derived from tweets and historic follow-through rate in the log. To capture the characteristics of different types of users, we partition the users into different groups and train the models separately.

3.2.1 Regression Model

In our framework, we use logistic regression model. In our case, the dependent variable is binary, i.e., similar or not. In logistic regression, the probabilities describing the possible outcomes are modeled as a logistic function of the explanatory variables as below.

$$p(Y|X) = \frac{1}{1 + e^{-(\alpha + \beta^T * X)}}$$

where X refers to the explanatory variables and α and β are the regression coefficients to be estimated.

To meet the needs of similarity computation for hundreds of millions of users, we implemented a regression model on a Hadoop-based, Pig-centric analytics platform at Twitter [27], where data sampling, feature generation, training, and testing can be accomplished directly in Pig, via carefully crafted loaders, storage functions, and user-defined functions.

To handle millions of data points for training, we use online learning algorithms and focus on stochastic gradient descent (SGD). Stochastic gradient descent is an approximation to gradient descent methods commonly used in batch learning. Specifically we use a Pegasos style SGD. Pegasos [29] was originally proposed for SVM, but it is applicable to other loss functions, such as the one due to Logistic Regression. After each iteration, a verification is performed to ensure that the weight vector is contained in an ℓ_2 ball determined by the regularization parameter.

3.2.2 Training Data

We construct the training data based on log data of "You might also want to follow" as shown in Section 2.2.1. In "You might also want to follow", if a user visits someone's Twitter profile page and follows that account, named as target user, similar accounts to the target user will pop up as suggestions. We log the user's follow behavior in a unified "client events" log format [23]. If a user follows the suggestion, we consider the (target user, suggestion) pair as an positive example. Otherwise, we consider it as negative.

As shown in Section 2.2, we have many products related to our similar-to framework. We choose the log of "You might also want to follow" for training for the following reasons. First, this product has a good amount of traffic, and provides plenty of training examples. Second, this product contains high fidelity data. We only pop up the similar-to suggestions whenever a user follow the account on its profile page. At that time the user has a follow intention and hence pays relatively more attention to the suggestions compared with other products. This is corroborated by the fact that this product has a much higher follow-through rate than other products driven by the same data.

3.2.3 Feature Selection

In our regression model, we have explored a variety of features in the feature selection process. We selected our used features iteratively. If a feature has a very small coefficient in regression model, we consider it not helpful so that we eliminate it from the model.

The following features are used in our regression model:

- CosineFollow score. It is the score from CosineFollow algorithm based on follow graph in candidate generation as shown Section 3.1.1.
- Number of suggestion's followers. The similar-to framework powers up many products at Twitter as shown in Section 2.2. In these products, the users are inclined to follow more popular accounts. The number of a user's followers is a good indication of the popularity.
- Pagerank score: the pagerank score of two similar accounts is also expected to be similar.
- Historic follow-through rate. A user's follow behavior is a valuable resource for similarity measure. For example, in "You might also want to follow", if a user follows a provided suggestion, it gives the hint that the user and suggestion might be similar. The follow-through rate of the similar-to products can be considered an effective feedback signal from the users.

The following features are examples of features that were explored and not found to be useful:

- Mutual follow. A mutual follow is a bidirectional edge in follow graph. The assumption was that if two users follow each other, they might be similar. However, it turns out this signal is noisy and not predictive. On one hand, two similar users are not necessarily likely to follow each other. For example, similar famous actors may not follow each other, since they are not interested in the other. On the other hand, if two users follow each other, they may or may not be similar. For example, the celebrities often follow their fans back, but they are not really similar to each other.
- Topics. For each user, topics are derived from the content using topic modeling methods such as LDA [4]. However, whether two users have similar topics of interest or not is not a powerful signal for user similarity. For example, it is difficult to claim two users are similar if both are interested in sports.
- Location. The geographical closeness of different granularities are experimented with, including city, Metropolitan Statistical Area (MSA) and country. It turns out that geographical tie are too loose for measuring similarity between users.
- Email domain. Email domain is a good resource to identify the user's affiliation. For example, if two users are from the same ".edu" domain, they are likely from the same school. Unfortunately the same school is still too coarse for similarity computation in most cases compared with other signals.

3.2.4 User Groups

We noticed that the features have significantly different predictive patterns for different types of users. For example, the historic follow-through rate is a very powerful signal for celebrities, while the CosineFollow score works much better for non-celebrity users. Hence we observed that having a single model across all the users was unable to capture the characteristics of different users. The solution we converged on to was to partition the users into different groups and train a separate model for each group.

There are many approaches to partitioning the users such as demographics. In this problem we find the influence of users is a good measure for partition, because users of various influence exhibit different similar-to chracterisitics. We first compute a measure for each user to calibrate its influence on Twitter. Similar to PageRank[5], we use the following assumption to compute influence measure: a user is influential if it is followed by other influential users. Then, we bucket users into 7 groups based on the ascending order of the calculated influence measure. Users in group 7 are the most influential ones, while those in group 1 are the least influential ones. [3]

4. EVALUATION

To demonstrate the effectiveness of our similar-to framework, we evaluate the performance in both quantitatively and qualitatively.

4.1 Quantitative Measures

4.1.1 Human Evaluation

In order to measure the precision of our framework, we use Amazon Mechanic Turk to label our similar-to results. For each user group in Section 3.2.4, we sampled 200 users and provided 5 sampled similar user accounts for human judges. We chose users whose Twitter language was English. We use sampling theory [2] to efficiently estimate the precision at 5, 10 and 20. In particular, we sample two from top 5 candidates, one more from top 6-10, one more from top 11-15 and one more from 16-20. For each (user, suggestion) pair, we provide three options for judges: Yes (the two users are similar), No (the two users are not similar), Hard to say (too few information, foreign language, deleted account, etc.). For each pair, we invite 3 judges and pick up the label using majority voting. We use Fleiss' kappa [9] to assess the reliability of judge agreement. Fleiss' kappa is 0.352 in the evaluation.

Also, to help the judges get started, we provided some examples of similar users. @KingJames is similar to @kobebryant, because both Kobe Bryant and LeBron James are NBA stars. @ladygaga is not similar to @kobebryant, because Kobe Bryant and Lady Gaga are in different fields though they are both celebrities. @LinkedIn is similar to @twitter, because both Twitter and LinkedIn are social network sites. @justinbieber is not similar to @twitter, because Justin Bieber is a singer and is in a different domain.

As shown in Section 3.2.4, we bucket the users into 7 groups based on the ascending order of the influence measure. We show the percentage of "Hard to say" in the judgment in Figure 5. For the groups consisting of influential users such as group 6 and group 7, there are only few pairs that the judges categorize as "Hard to say", because these users are famous and easy to judge the similarities. On the other hand, for group 1, i.e., the least influential users, it is more difficult for the judges to evaluate the similarity, because the users are less popular and have few information so that the similarity is more ambiguous.

[3]We also tried adding the user buckets as a feature in the single model, but having different models outperformed that.

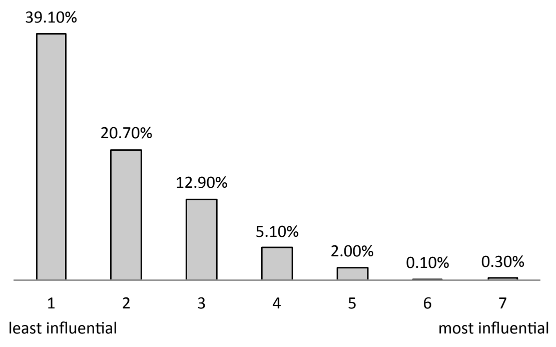


Figure 5: Percentage of "Hard to say" for different user groups

In Table 1, we compare the precision of CosineFollow algorithm and similar-to framework across different user groups. From the data, we can see that the result of similar-to framework outperforms the one of vanilla CosineFollow by a significant margin.

4.1.2 Twitter Products Evaluation

A more direct evaluation of the performance is to apply our similar-to framework's result to Twitter products. We use follow-through rate as the measure. Follow-though rate is the percentage of the follows among all the impressions of the suggestions. We experimented in three products including "You might also want to follow" (see Section 2.2.1, similar-to module on Iphone and Android platforms (see Section 2.2.2. We compared the result of our similar-to framework with the existing data set that is the output of the CosineFollow algorithm. We observed that our follow-through rate increased by 38% in "You might also want to follow", by 32% in "similar-to" module on iPhone and from by 29% in "similar-to" module on Android platform. The significant improvement of follow-through rate converted to millions of additional follow edges every day.

4.2 Case Study

Besides the quantitative evaluation, we also provide a case study to evaluate the performance of our similar-to framework in a more qualitative manner. We take @kobebryant as an example, and compare the results returned by CosineFollow algorithm and the similar-to framework in Table 2. CosineFollow algorithm is based on follow graph, so the similar users provided by CosineFollow have many common followers. In the CosineFollow result, we find not only some NBA players but also some reporters covering NBA. However, they might not be relevant to Kobe Byrant particularly. Our similar-to framework consolidates signals from multiple sources, so the result is more comprehensive and much better. In similar-to framework, we not only discover Kobe Bryant's teammates at Lakers including @DwightHoward, @SteveNash and @paugasol, but also identify @KingJames as the top similar candidate. In additional, similar-to framework is able to detect @Lakers and @NBA, which are the official accounts for Lakers and NBA respectively.

5. CONCLUSIONS

Table 1: Precision for different user groups in human evaluation

Group	CosineFollow Algorithm				Similar-to Framework			
	P@5	P@10	P@15	P@20	P@5	P@10	P@15	P@20
1	15.0%	13.0%	12.0%	11.8%	**16.5%**	**16.3%**	**15.8%**	**15.6%**
2	19.5%	19.3%	20.5%	19.9%	**23.0%**	**22.0%**	**23.7%**	**23.0%**
3	36.5%	35.3%	35.5%	34.1%	**37.5%**	**37.8%**	**37.5%**	**35.9%**
4	55.5%	57.3%	57.5%	57.9%	**70.5%**	**67.3%**	**67.5%**	**67.1%**
5	77.5%	78.3%	78.2%	77.1%	**84.0%**	**81.0%**	**80.7%**	**81.0%**
6	**89.0%**	**90.0%**	**88.0%**	**88.5%**	87.5%	84.8%	83.5%	83.4%
7	88.5%	87.8%	85.8%	83.9%	**88.5%**	**88.8%**	**87.8%**	**86.9%**
Average	54.5%	54.4%	53.9%	53.3%	**58.2%**	**56.8%**	**56.6%**	**56.1%**

Table 2: Similar users to @kobebryant

CosineFollow Algorithm

Rank	Twitter @handler	Display name	Score
1	@KDTrey5	Kevin Durant	0.220
2	@stephenasmith	Stephen A Smith	0.213
3	@LakersReporter	Mike Trudell	0.174
4	@MagicJohnson	Earvin Magic Johnson	0.174
5	@MettaWorldPeace	Metta World Peace	0.170
6	@SteveNash	Steve Nash	0.160
7	@jadande	J.A. Adande	0.156
8	@DeronWilliams	Deron Williams	0.151
9	@RajonRondo	Rajon Rondo	0.144
10	@WojYahooNBA	Adrian Wojnarowski	0.142

Similar-to Framework

Rank	Twitter @handler	Display name	Score
1	@KingJames	LeBron James	0.138
2	@DwightHoward	Dwight Howard	0.126
3	@Lakers	Los Angeles Lakers	0.121
4	@CP3	Chris Paul	0.115
5	@KDTrey5	Kevin Durant	0.114
6	@NBA	NBA	0.110
7	@SteveNash	Steve Nash	0.109
8	@paugasol	Pau Gasol	0.108
9	@carmeloanthony	Carmelo Anthony	0.107
10	@blakegriffin32	Blake Griffin	0.106

In this paper we present the similar-to framework at Twitter. The similar-to problem at Twitter is challenging because of the scalability issue posed by the large number of active users and the need for consolidating the various signals in an effective and scalable manner. We proposed a machine learning based framework built on Hadoop, which is capable of discovering similar accounts of high quality for hundreds of millions of users on a daily basis. In the framework, we first construct a candidate set via a graph-based cosine similarity algorithm, and then re-rank the candidates based on various signals using a logistic regression model trained on historic follow data from Twitter similar-to products. A human evaluation and experiments in real Twitter product show the effectiveness of the framework.

6. REFERENCES

[1] A. Anderson, D. P. Huttenlocher, J. M. Kleinberg, and J. Leskovec. Effects of user similarity in social media. In *WSDM*, pages 703–712, 2012.

[2] J. A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In *SIGIR*, pages 541–548, 2006.

[3] L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In *WSDM*, pages 635–644, 2011.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[6] H. Cheng, Y. Zhou, and J. X. Yu. Clustering large attributed graphs: A balance between structural and attribute similarities. *TKDD*, 5(2):12, 2011.

[7] S. Debnath, N. Ganguly, and P. Mitra. Feature weighting in content based recommendation system using social network analysis. In *WWW*, pages 1041–1042, 2008.

[8] P. Flajolet, E. Fusy, O. Gandouet, and et al. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In *AOFA*, 2007.

[9] J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76:378–382, 1971.

[10] D. Fogaras and B. Rácz. Scaling link-based similarity search. In *WWW*, pages 641–650, 2005.

[11] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. Knowl. Data Eng.*, 19(3):355–369, 2007.

[12] A. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava. Building a highlevel dataflow system on top of mapreduce: The pig experience. *PVLDB*, 2(2):1414–1425, 2009.

[13] L. Getoor. Link mining: a new data mining challenge. *SIGKDD Explorations*, 5(1):84–89, 2003.

[14] L. Getoor and C. P. Diehl. Link mining: a survey. *SIGKDD Explorations*, 7(2):3–12, 2005.

[15] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh. Wtf: the who to follow service at twitter. In *WWW*, pages 505–514, 2013.

[16] M. A. Hasan and M. J. Zaki. A survey of link prediction in social networks. In *Social Network Data Analytics*, pages 243–275. Springer, 2011.

[17] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of

dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[18] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *KDD*, pages 538–543, 2002.

[19] H. Kashima and N. Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *ICDM*, pages 340–349, 2006.

[20] H. Kashima, T. Kato, Y. Yamanishi, M. Sugiyama, and K. Tsuda. Link propagation: A fast semi-supervised learning algorithm for link prediction. In *SDM*, pages 1099–1110, 2009.

[21] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *TKDD*, 4(1), 2010.

[22] J. Kunegis and A. Lommatzsch. Learning spectral graph transformations for link prediction. In *ICML*, page 71, 2009.

[23] G. Lee, J. Lin, C. Liu, A. Lorek, and D. V. Ryaboy. The unified logging infrastructure for data analytics at twitter. *PVLDB*, 5(12):1771–1780, 2012.

[24] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg. Predicting positive and negative links in online social networks. In *WWW*, pages 641–650, 2010.

[25] D. Liben-Nowell and J. M. Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559, 2003.

[26] D. Liben-Nowell and J. M. Kleinberg. The link-prediction problem for social networks. *JASIST*, 58(7):1019–1031, 2007.

[27] J. Lin and A. Kolcz. Large-scale machine learning at twitter. In *SIGMOD Conference*, pages 793–804, 2012.

[28] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig latin: a not-so-foreign language for data processing. In *SIGMOD Conference*, pages 1099–1110, 2008.

[29] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*, pages 807–814, 2007.

[30] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 4(11):992–1003, 2011.

[31] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu. Rankclus: integrating clustering with ranking for heterogeneous information network analysis. In *EDBT*, pages 565–576, 2009.

[32] B. Taskar, M. F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *NIPS*, 2003.

[33] X. Yin, J. Han, and P. S. Yu. Linkclus: Efficient clustering via heterogeneous semantic links. In *VLDB*, pages 427–438, 2006.

[34] Z. Yin, M. Gupta, T. Weninger, and J. Han. Linkrec: a unified framework for link recommendation with user attributes and graph structure. In *WWW*, pages 1211–1212, 2010.

[35] Z. Yin, M. Gupta, T. Weninger, and J. Han. A unified framework for link recommendation using random walks. In *ASONAM*, pages 152–159, 2010.

[36] R. B. Zadeh and A. Goel. Dimension independent similarity computation. *CoRR*, abs/1206.2082, 2012.

[37] P. Zhao, J. Han, and Y. Sun. P-rank: a comprehensive structural similarity measure over information networks. In *CIKM*, pages 553–562, 2009.

[38] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.