The Twitaholic Next Door.

Scalable friend recommender system using a concept-sensitive hash function

Patrick Bamba
Université Jean Monnet
25, rue du Dr Rémy Annino
F-42000, Saint-Etienne, France
patrick.bamba@ymail.com

Julien Subercaze
Université Jean Monnet
25, rue du Dr Rémy Annino
F-42000, Saint-Etienne, France
julien.subercaze@univ-st-etienne.fr

Christophe Gravier
Université Jean Monnet
25, rue du Dr Rémy Annino
F-42000, Saint-Etienne, France
christophe.gravier@univ-st-etienne.fr

Nabil Benmira
Graphinium - ESSEC VENTURES
CNIT - BP230
92053 Paris La Défense
nbenmira@graphinium.com

Jimi Fontaine
Graphinium - ESSEC VENTURES
CNIT - BP230
92053 Paris La Défense
jfontaine@graphinium.com

ABSTRACT

In this paper we present a Friend Recommender System for micro-blogging. Traditional batch processing of massive amounts of data makes it difficult to provide a near-real time friend recommender system or even a system that can properly scale to millions of users. In order to overcome these issues, we have designed a solution that represents user-generated micro posts as a set of pseudo-cliques. These graphs are assigned a hash value using an original Concept-Sensitive Hash function, a new sub-kind of Locally-Sensitive Hash functions. Finally, since the user profiles are represented as a binary footprint, the pairwise comparison of footprints using the Hamming distance provides scalability to the recommender system. The paper goes with an online application relying on a large Twitter dataset, so that the reader can freely experiment the system.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; E.1 [**Data Structures**]: Graphs and networks

General Terms

Theory, Algorithms

Keywords

Social Networks, Twitter, Friends, Recommender System, Graph, Pseudo-Clique, Locally-Sensitive Hash.

1. INTRODUCTION

Microblogging websites produce a tremendous amount of data each second. These Web 2.0 services are displaying an exponential growth of human-generated data over time. For instance, Twitter was known to publish an average of 140 millions tweets per day as of march 2011[1]. In a single year, this number has increased up to 340 millions tweets[2]. Consequently, near real-time recommendation systems require very scalable algorithms to proceed such a massive amount of data. Twitter represents a pool of 140 millions active users in March 2012. A primary need for Twitter users is therefore to identify a dozen of possible friends among these millions of users. Recommending new friends to Twitter users is therefore twofold: a friend recommender system stands as an algorithmic problem, and scalability is compulsory. Our goal is to provide an efficient and scalable content-based friend recommender system for Twitter users that scales for millions of users and tweets on few commodity servers. A content-based friend recommender system aims at drawing recommendations out of users' tweets and not from their social graph. We expect users to be more interested in other users sharing the same interests, and not necessarily by users who are only sharing friend-of-a-friend relationships in the Twitter social user graph. Related works are summarized in Section 2. To achieve our goal, our system proceeds in two major steps. Section 3 presents the first step of the system which is the encoding of the user profile as a pseudo-clique. Section 4 explains the second step in which the system creates a binary footprint of the pseudo-clique that represents the user profile. In Section 5 we introduce the system architecture and implementation. Finally, Section 6 concludes.

2. RELATED WORKS

Content recommender systems recommend tweets or external content sources to users whereas users recommender

[1]`http://blog.twitter.com/2011/03/numbers.html`
[2]`http://blog.twitter.com/2012/03/twitter-turns-six.html`

systems predict links in a social graph. Both types of recommenders rely on a characterization of the user. For example [12] is using Twitter for topical news recommendation. The system crawls Twitter and RSS feeds, indexes terms from both sources (as well as user's friends on Twitter) using Lucene's TF-IDF library and provides recommendation based either on the user's public timeline or friends' timelines matching with the article terms. Regarding user recommender, *Twopics* [9] uses Wikipedia to query for named entities in tweets in order to build a topic profile. Even if no performance analysis is provided, [5] showed that executing SPARQL queries over the web of linked data takes at least 20 seconds even with all data locally retrieved in advance, which discards *de facto* such an approach for real-time purpose. Given the hindrance due to the Web of Data speaking of performance at query time, only a single document approach can be used when scalability is at stake. The bag of words approach to characterize users or documents (as used in [12, 4]) has shown its limitations, and machine learning techniques have been developed to go beyond bag-of-words representation. The most popular are continuous Conditional Random Fields [7] and Latent Dirichlet Association (LDA) [1]. The main drawback of these machine learning techniques is the learning part, which is prohibitively extensive for real-time processing. This explains why the online learning paradigm is gaining momentum in the Machine Learning community. In this work, we investigate whether original document-centric approach that makes use of statistics and graph techniques could still scale while preserving the advantage to exploit Semantic relatedness between terms in documents.

3. STEP 1 : BUILDING USER PROFILES

In order to provide the most efficient user profile from tweets, we applied standard text pre-processing strategies (tokenization and stop words deletion). Then the next step is inspired by the paper of Matsuo & Ishizuka [8] on keywords extraction. They present a method to extract keywords from a single document using statistical information. In a first step their algorithm computes the co-occurrence matrix of the terms in each sentence of the document. Frequent terms are counted, and then clustered by pairs according to a threshold. The most representative term of the cluster item serves as a representative for terms in the same cluster. Instead of clustering terms in a pairwise manner as proposed in [8], we opted for a graph representation. We look for clusters of terms in order to model the user-generated content. In our work, each term (one of the graph edges) is given a weight, which is the "freshness" associated to this cluster. It is an age indication, calculated as the current time of computation minus the date of last use of the term. This information is normalized accross all terms employed by a given user.

To sum up, our algorithm characterizes the user by extracting the meaningful clusters from his sequence of tweets and encode this as a graph. We model the document extraction result as a graph. As a consequence, the problem of measuring semantic relatedness is to look for clusters in the graph. From a graph point of view, pairwise clustering from [8] is equivalent to the search for connected components in the graph of terms. In graph theory, a maximal complete subgraph is called a clique (i.e. a clique is a subgraph with a density equal to 1). The problem of finding cliques in a

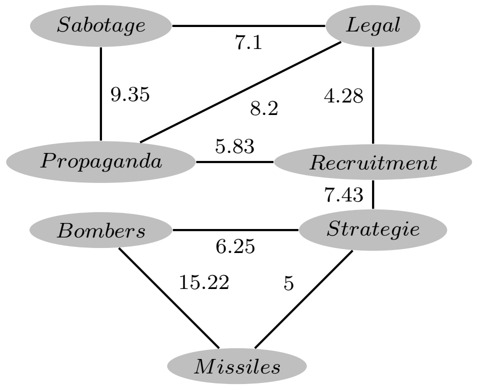


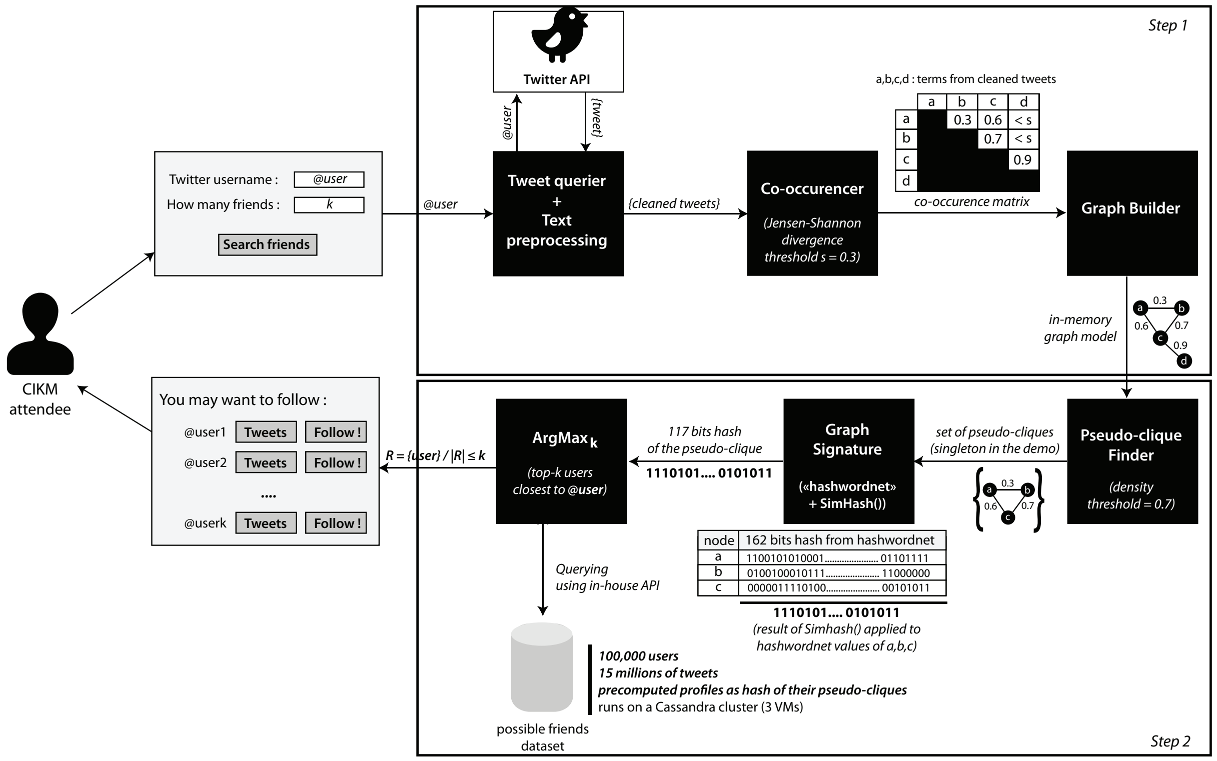
Figure 1: Graph of terms with weights computed using Jensen-Shannon Divergence distance.

graph is NP-Complete and the current most efficient algorithm is the one from Bron-Kerbosch [2]. Our intuition is that pseudo-cliques may offer a more flexible manner to detect interesting clusters in the graph of terms, as the one presented in Figure 1. A pseudo-clique is a subgraph whose density is greater than a fixed threshold σ. Any clique is also a pseudo-clique. For instance, in Figure 1, `{Bombers, Strategie, Missiles}` is a clique, and `{Sabotage, Legal, Propadanda, Recruitment}` is a pseudo-clique of density equals to $\frac{5}{6}$. We have designed an algorithm to detect pseudo-cliques from the graph representation of the co-occurrence matrix. This algorithm performs a most connected nodes traversal approach, with a test that is reduced to the density of the current clique. The result of this algorithm is a set of pseudo-cliques. For instance, based on the graph 1, our algorithm would represent the user with the set of pseudo-cliques `{{Sabotage, Legal, Propadanda, Recruitment}`, `{Bombers, Strategie, Missiles}}`.

In summary, the main operations for the first step of our system presented so far are the following: i) Preprocess the tweets text, ii) Build the graph of terms from the co-occurrence matrix,and finally iii) Find relevant pseudo-cliques in the graph.

4. STEP 2 : HASHING USER PROFILES

Each user can have his/her profile modeled as a set of pseudo-cliques, whose vertices are terms extracted from the user's tweets. In order to find possible friends for a given user, we need a distance metric between two sets of pseudo-cliques. For the sake of the explanation, we assume in the following that the profile of a user is composed of a single pseudo-clique. This is also how the system is currently implemented for the presented online application (more discussion on this at Section 6). In order to satisfy the near-real time and scalablity objective, our intuition was to look for a higher dimensional distance metric for computing similarities between two pseudo-cliques. We therefore investigated on the different options to encode a pseudo-clique as a bit array, so that the Hamming distance could be used as a similarity metric between user profiles. This issue is related to the field of graph hash functions. An important review and some approaches can be found in [11]. However, we aim at a hash function that would preserve likelihood of the pseudo-



	a	b	c	d
a		0.3	0.6	< s
b			0.7	< s
c				0.9
d				

node	162 bits hash from hashwordnet
a	1100101010001..................... 01101111
b	0100100010111..................... 11000000
c	0000011110100..................... 00101011

Figure 2: Architecture of the software solution

cliques when they are represented as hash values. Using a cryptographic hash function to a pseudo-clique would only result in obtaining possibly completely different hash values for two near-similar pseudo-cliques. This is because a cryptographic hash function needs to avoid preserving distances between original data and encoded data. Unlike cryptographic hash functions, Locally Sensitive Hash (LSH) functions[6] aim at preserving likelihood between original data and their corresponding hash values. Our intuition is to make use of a LSH function to hash terms in the graph of terms, and then to generate a footprint from the graph of hashs. [3] proposed SimHash, a hash function for generating a footprint out of a graph. SimHash can be applied to any kind of resource (document, images . . .), and in our case a graph. In SimHash, the resource, usually a document, is splitted into token, possibly weighted. Each token is then represented as its hash value, as the result of a traditional cryptographic function applied to the token, which is originally a string. Then, a vector V, of length of the desired hash size, is initialized to 0. For each hash value for the set of tokens, the i^{th} element of V is *decreased* by the corresponding token's weight if the i^{th} bit of the hash value is 0. Otherwise, the i^{th} element of V is *increased* by the corresponding token's weight. SimHash works well even for small fingerprints [3]. Nonetheless, the use of a traditional function on string for obtaining hash values of a token, restricts its usage to the detection of near duplicates. SimHash was historically applied to the detection of near-duplicates of webpages. It is possible to use SimHash in order to make a footprint of a pseudo-clique with, given our user profile representation, the following settings:

- As SimHash features : the set of edges and vertices of the pseudo-clique serve as our set of tokens,
- As Simhash weights :
  - For vertices: the freshness information (introduced at Section 3) which is the node's weight,
  - For edges : the normalized Jensen-Shannon diergence values, which is the edge's weight.

This process would allow us to detect near-duplicates in pseudo-cliques (i.e. user profiles) using the Hamming distance of the generated footprint out of the SimHash function applied to each user's pseudo-clique.

However, this solution presents a main drawback: users who are denoting the same concepts in their tweets but using different terms, would not present profile with a high degree of similarity. The main hindrance to this is the usage of a traditional hash function for obtaining hash values of a token. This is because the hash function of the token conveys the similarity between other documents since the same token will have the same hash function. In case the two pseudo-clique contain terms related to the same concepts but represented with different strings, the hash function will produce completely different hash values. Hence, the resulting footprint would have lost the conceptual relationship between the two terms. In order to insert a concept-sensitive hash function in our system, we need to opt for a hash function that would take into account conceptual similarity of the tokens. The hash function we have therefore introduced in our system works as follows. We rely on the Wordnet dictionary [10]. Wordnet contains hyponymy and synonymy relationships between concepts, which are represented by a bag of terms in English. We created a hash function that produces a hash value from a term depending on its location in the Wordnet hypernym tree. In order to do so, we built a reverse table of Wordnet that associate a hash value to Wordnet nodes. The hash value is constructed by performing a depth-first search of Wordnet entries, and tagging each entry with a unique and minimal binary value, whose length is determined by the number of siblings for each entry. In order to uniquely encode all concepts, 117 bits are needed. In other words, we associated a unique hash value of 117 bits for each node in Wordnet, and store this in an indexed table. We called this table "hashwordnet". This table was constructed once and is shipped in the server-side friend recommender software presented at Section 5. Consequently, each term in the pseudo-clique is associated with a unique hash from this table. Two terms belonging to the same concept in Wordnet will have the same hash values. Furthermore, two terms whose associated concepts are closely similar in Wordnet will be represented by similar hash values. When the SimHash function is applied, it will be sensitive to this similarity of token's hash values, instead of only relying on the strict equality of strings between tokens. The footprint can be expected to convey the Wordnet concepts information to which the terms in the pseudo-clique refer to.

5. IMPLEMENTATION

The complete architecture of the system presented in the paper is provided at Figure 2. We have implemented the entire software architecture described earlier in this paper on a server-side component in the Java programming language. The system relies on a dataset that consists of 15,000,000 tweets issued by 150,000 users. The dataset also have the precomputed user profiles, which means the footprint of each user encoded from their tweets in the dataset, and according to our concept-sensitive hash function. It is hosted on a cluster of three Apache Cassandra[3] instances in our laboratory private infrastructure.

A Web interface asks from the conference attendee its Twitter account name and the number k of friend recommendation that the end-user whishes. After the Twitter secure authentication through Twitter OAuth[4] service, the server-side component retrieves the end-user's tweets. It constructs the end-user's profile, which means it goes through all the steps as illustrated at Figure 2. Finally, the Web application displays the sorted list of the k closest users from our dataset, whose footprint are the closest to the end-user's footprint according to a Hamming distance computation. Because the end-user's tweets will be processed in seconds, and a recommendation is made out of two millions profiles, this demonstrates the scalability of the solution. The application is accessible online for anyone to experiment the solution on his own account(s)[5].

6. CONCLUSION

This paper introduced a new scalable and near real-time approach for friend recommendation on Twitter. The friend recommender system works as follows: users profiles are represented as a pseudo clique, this pseudo-clique is encoded as a graph, whose nodes are hashed using an original concept-sensitive hash function introduced in this paper. Using this function, the more two terms are similar using Wordnet as a distance through the hyponymy relation, the more their two hash values will be similar. A binary footprint is constructed from this pseudo-clique using the SimHash algorithm. The comparison of one user footprint against all other precomputed footprints from the other users makes the system scalable at runtime.

In future works, we expect to represent users profiles as a set of pseudo-cliques. The key issue relies in generating a footprint not of a single pseudo-clique, but out of several pseudo-cliques.

7. REFERENCES

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

[2] K. J. Bron, C. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16:575–577, 1973.

[3] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. *Approximation Algorithms for Combinatorial Optimization*, pages 139–152, 2000.

[4] J. Hannon, M. Bennett, and B. Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 199–206. ACM, 2010.

[5] O. Hartig, C. Bizer, and J. Freytag. Executing sparql queries over the web of linked data. *The Semantic Web-ISWC 2009*, pages 293–309, 2009.

[6] P. Indyk and A. Gionis. Similarity search in high dimensions via hashing. *Proceedings of the 25th VLDB*, pages 518–529, 1999.

[7] J. Lafferty, A. McCallum, and F. Pereira. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*, volume CONF 18, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[8] Y. Matsuo and M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(1):157–170, 2004.

[9] M. Michelson and S. Macskassy. Discovering users' topics of interest on twitter: a first look. In *Proceedings of the fourth workshop on Analytics for noisy unstructured text data*, pages 73–80. ACM, 2010.

[10] G. A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41, 1995.

[11] C. Papadimitriou. Wordnet: A lexical database for english. *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 749–753, 2001.

[12] O. Phelan, K. McCarthy, and B. Smyth. Using twitter to recommend real-time topical news. In *Proceedings of the third ACM conference on Recommender systems*, pages 385–388. ACM, 2009.

[3]http://cassandra.apache.org/
[4]http://oauth.net/
[5]http://demo-satin.telecom-st-etienne.fr/lshrecommender/