Real-Time Community Detection in Large Social Networks on a Laptop

Ben Chamberlain
Department of Computing
Imperial College London
London, UK

Clive Humby
Starcount
2 Riding House Street
London, UK

Marc Peter Deisenroth
Department of Computing
Imperial College London
London, UK

ABSTRACT

For a broad range of research, governmental and commercial applications, it is important to understand the allegiances, communities and structure of key players in society. One promising direction towards extracting this information is to exploit the rich relational data in digital social networks (the social graph). As social media data sets are very large, most approaches make use of distributed computing systems for this purpose. Distributing graph processing requires solving many difficult engineering problems, which has lead some researchers to look at single-machine solutions that are faster and easier to maintain. In this article, we present a single-machine real-time system for large-scale graph processing that allows analysts to interactively explore graph structures. The key idea is that the aggregate actions of large numbers of users can be compressed into a data structure that encapsulates user similarities while being robust to noise and queryable in real-time. We achieve single machine real-time performance by compressing the neighbourhood graph of each vertex using minhash signatures and facilitate rapid queries through Locality Sensitive Hashing. These techniques reduce query times from hours using industrial desktop machines operating on the full graph to milliseconds on standard laptops. Our method allows exploration of strongly associated regions (i.e. communities) of large graphs in real-time on a laptop. It has been deployed in software that is actively used by analysts to understand social network structure.

1. INTRODUCTION

Algorithms to discover groups of associated entities from relational (networked) data sets are often called *community detection* methods. They come in two forms: global methods, which partition the entire graph, and local methods that look for vertices that are related to an input vertex and only work on a small part of the graph. We are concerned with community detection on large graphs that runs on a single commodity computer. To achieve this we combine the two approaches using local community detection to identify an interesting region of a graph and then applying global community detection to help understand that region.

Applications include: security, where analysts explore a network looking for groups of potential adversaries; social sciences, where queries can establish the important relationships between individuals of interest; e-commerce, where queries reveal related products or users; marketing, where companies seek to optimise advertising channels or celebrity endorsement portfolios. To meet these emergent needs analysts must explore vast networks. Our work demonstrates an alternative monetisation channel for Digital Social Networks (DSN)s; that social media data can be used in multiple industries for decision support. Decision support does not disrupt user experience in the way that sponsored links or feed advertising do. It offers another channel for social media providers to monetise their data, helping them to continue to provide free services that are valued by billions of people globally.

As a sketch of how the system is used we imagine a company that wants to enter a new foreign market. To do this they need to understand the market's competitors, customers and marketing channels. Using our system they could input the Twitter handles for their existing products, key people, brands and endorsers and in real-time receive the most similar accounts to their company in that market. The output is structured into groups such as magazines, sportspeople, companies in the same industry etc. They could then examine the results and explore different regions by changing the input accounts.

Throughout this paper we refer to graphs. In this context a graph is a collection of vertices (social media accounts) and edges connecting them. We are particularly interested in the neighbourhood graph, which is often used to compare vertices and has been shown to have attractive properties in this regard [24, 23]. The neighbourhood graph of a vertex consists of the set of all vertices that are directly connected to it, irrespective of the edge direction. We propose that robust associations between social network accounts can be reached by considering the similarity of their neighbourhood graphs. This proposition relies on the existence of homophily in social networks. The homophily principle states that people with similar attributes are more likely to form relationships [16]. Accordingly social media accounts with similar neighbourhood graphs are likely to have similar attributes.

We seek to build a system that is robust to failure, does not require engineering support and is parsimonious with the time of its users. The first two requirements lead us to search for a single machine solution and the later requirement prescribes a real-time (or as close as possible) system. To produce a real-time single-machine system that operates on large graphs we must solve two problems: (1) Fit the graph in the memory of a single (commodity) machine; (2) Calculate the similarity of neighbourhood graphs containing up to 100m verices in milliseconds. We address

these problems by compressing the neighbourhood graphs into fixed-length minhash signatures. Minhash signatures vastly reduce the size of the graph while encoding an efficient estimation of the Jaccard similarity between any two neighbourhoods. Choosing appropriate length minhash signatures squeezes the graph into memory. To achieve real-time querying we use the elements of the minhash signatures to build a Locality Sensitive Hashing (LSH) data structure. LSH facilitates querying of similar accounts in constant time. A combination of minhashing and LSH allows analysts to enter a set of accounts and receive the set of most related accounts in milliseconds. Our contributions are:

1. We establish that robust associations between social media users can be determined by means of the Jaccard similarity of their neighbourhood graphs.

2. We show that the approximations implicit in minhashing and LSH minimally degrade performance and allow querying of very large graphs in real time.

3. System design and evaluation: We have designed and evaluated an end-to-end Python system for extracting data from social media providers, compressing the data into a form where it can be efficiently queried in real time.

4. We demonstrate how queries can be applied to a range of problems in graph analysis, e.g., understanding the structure of industries, allegiances within political parties and the public image of a brand.

2. DATA

In this article, we focus on Twitter data because Twitter is the most widely used DSN for academic research. The Twitter Follower graph consists of roughly one billion vertices and 30 billion edges. To show that our method generalises to other social networks, we also present results using a Facebook Pages engagement graph containing 450 million vertices and 700 million edges (see Section 5).

To collect Twitter data we use the REST API to crawl the network identifying every account with more than 10,000 Followers[1] and gather their complete follower lists. To optimize data throughput while remaining within the DSN rate limits we developed an asynchronous distributed network crawler using Python's Twisted library [26]. Our data set contains 675,000 such accounts with a total of 1.5×10^{10} Followers, of which 7×10^8 were unique. We restrict the data set to accounts with greater than 10,000 Followers (though 700 million Twitter accounts are used to build the signatures) because accounts that are smaller than this are unlikely to be informative to end users.

To generate data from Facebook we matched the Twitter accounts with greater than 10,000 Followers to Facebook Page accounts [2] using a combination of automatic account name matching and manual verification. Facebook Page likes are not available retrospectively, but can be collected through a real-time stream. We collected the stream over a period of two years from late 2013.

[1]The number of Followers is contained in the Twitter account metadata, i.e., it's available without collecting and counting all of the edges.

[2]Facebook pages are the public equivalent of the private profiles. Many influential users have a Facebook Page.

3. RELATED WORK

Existing approaches to large scale, efficient, community detection have three flavours: More efficient community detection algorithms, innovative ways to perform processing on large graphs and data structures for graph compression and search. Table 1 shows related approaches to this problem and which constraints they satisfy.

Table 1: Comparison of related work. SCM is Single Commodity Machine and MO is Modularity Optimisation

Method	Real-time	Large graphs	SCM
MO [18]	✗	✗	✓
WALKTRAP [20]	✗	✗	✓
INFOMAP [22]	✗	✗	✓
Louvain method [3]	✗	✓	✓
BigClam [27]	✗	✓	✓
Graphci [14]	✗	✓	✓
Twitter WTF [9]	✓	✓	✗
Our Method	✓	✓	✓

There is an enormous community detection literature and Fortunato [8] provides an excellent overview. Approaches can be broadly categorised into local and global methods.

Global methods assign every vertex to a community, usually by partitioning the vertices. Many highly innovative schemes have been developed to do this [18, 22, 20]. Most global methods do not easily scale to very large data sets. The availability of data from the Web, DSNs and services like Wikipedia led to a focus on methods for large graphs [3, 27]. However, there are no real-time algorithms that could facilitate interactive exploration of social networks.

Unlike global community detection methods, local algorithms do not assign every vertex to a community. Instead they find vertices in the same community as a set of input vertices (seeds). For this reason they are normally faster than global methods. Kloumann et al. [13] conducted a comprehensive assessment of local community detection algorithms on large graphs and identified Personal PageRank (PPR) [11] as the clear winner. Recent advances have shown that seeding PPR with the neighbourhood graph can improve performance [24] and that PPR can be used to initiate *local* spectral methods with good results [25]. Random walk methods are usually evaluated on distributed computing resources (e.g., Hadoop). While distributed systems are continually improving, they are not always available to analysts, require skilled operators and have a typical overhead of several minutes per query.

A complimentary approach to efficient community detection is to develop more efficient and robust graph processing systems. Graphci is a single-machine system that offers a powerful and efficient alternative to processing large graphs [14]. Twitter also use a single-machine recommendation system that serves "Who To Follow (WTF)" recommendations across their entire user base [9]. WTF achieves this by loading the entire Twitter graph into memory. Following their design specification of 5 bytes per edge $5 \times 30 \times 10^9 =$ 150 GB of RAM would be required to load the current graph, which is an order of magnitude more than available on our target platforms.

The alternative to using large servers, clusters or disk storage for processing on large graphs is to compress the graph. Graph compression techniques were originally motivated by the desire for single machine processing on the Web Graph [1, 4] and have been adapted for social networks [7]. Such

system	runtime (s)	space (GB)
naive edge list	8000	240
minhash signatures	1	4
LSH with minhash	0.25	5

Table 2: Typical runtimes and space requirements for systems performing local community detection on the Twitter Follower network of 700m vertices and 20 billion edges and producing 100 vertex output communities

techniques achieve remarkable compression factors, but at the cost of slower access [9]. Minhashing is a technique for representing large sets with fixed length signatures that encode an estimate of the similarity between the original sets. When the sets are sub-graphs minhashing can be used for lossy graph compression. The pioneering work on minhashing was by [5]. Minhashing has been applied to clustering the Web by modelling each web page as a bag of words and building hashes from the count vectors [10]. Two important innovations that improve upon minhashing are b-Bit minhashing [15] and Odd Sketches [17]. Locality Sensitive Hashing (LSH) is a technique introduced by Indyk [12] for rapidly finding approximate near neighbours in high dimensional spaces. Recent LSH research improves on the original implementation by using the structure of the data to design better hash functions at the expense of being able to handle new data points [2].

4. REAL-TIME ASSOCIATION MINING

In this section, we detail our approach to real-time association mining in large social networks. Our method consists of two main stages: In stage one, we take a set of seed accounts and expand this set to a larger group containing the most related accounts to the seeds. Stage one uses a very fast nearest neighbour search. In stage two, we embed the results of stage one into a weighted graph where each edge is weighted by the Jaccard similarity of the two accounts it connects. We apply a global community detection algorithm to the weighted graph and visualise the results.

In the remainder of the paper we use the following notation: The i^{th} user account (or interchangeably, vertex of the network) is denoted by A_i and $N(A_i)$ gives the set of all accounts directly connected to A_i (the neighbours of A_i). The set of accounts that are input by a user into the system are called seeds and denoted by $S = \{A_1, A_2, ..., A_m\}$ while $C = \{A_1, A_2, ..., A_n\}$ (community) is used for the set of accounts that are returned by stage one of the process.

4.1 Stage 1: Seed Expansion

The first stage of the process takes a set of seed accounts as input, orders all other accounts by similarity to the seeds and outputs an expanded set of similar accounts. We require three ingredients:

1. A similarity metric between accounts
2. An efficient system for finding similar accounts
3. A stopping criterion to determine the number of accounts to return

4.1.1 Similarity Metric

To compare the similarity of any two vertices we use the neighbourhood graph. The neighbourhood graph of a vertex consists of the set of all vertices that are directly connected to it, irrespective of the edge direction. We propose that robust associations between social network accounts can be reached by considering the similarity of their neighbourhood graphs. As a similarity metric we choose the Jaccard similarity of neighbourhood graphs. We use the Jaccard similarity because it is a widely used metric to compare two sets and minhashing can be used to provide an unbiased estimator of the Jaccard similarity that is both time and space efficient. The Jaccard similarity is given by

$$J(A_i, A_j) = \frac{|N(A_i) \cap N(A_j)|}{|N(A_i) \cup N(A_j)|}, \quad (1)$$

where $N(A_i)$ is the set of neighbours of i^{th} account.

4.1.2 Efficient Account Search

To efficiently search for accounts that are similar to a set of seeds we represent every account as a minhash signature and use a Locality Sensitive Hashing (LSH) data structure based on the minhash signatures for approximate nearest neighbour search.

Rapid Jaccard Estimation via Minhash Signatures

Computing the Jaccard similarities in (1) is very expensive as each set can have up to 10^8 members and calculating intersections is super-linear. Multiple large intersection calculations can not be processed in real-time. There are two alternatives: either the Jaccard similarities can be precomputed for all possible pairs of vertices, or they can be estimated. Using pre-computed values for $n = 675,000$ would require caching $\frac{1}{2}n(n-1) \approx 2.5 \times 10^{11}$ floating point values, which is approximately 1TB and so not possible using commodity hardware. Therefore an estimation procedure is required.

The minhashing compression technique of [6] generates unbiased estimates of the Jaccard similarity $\hat{J}$ in $O(K)$, where K is the number of hash functions in the signature.

$$\hat{J}(A_i, A_j) = I/K, \quad (2)$$

where we define

$$I = \sum_{k=1}^{K} \delta(h_k(A_i), h_k(A_j)), \quad (3)$$

$$\delta(h_k(A_i), h_k(A_j)) = \begin{cases} 1 & \text{if } h_k(A_i) = h_k(A_j) \\ 0 & \text{if } h_k(A_i) \neq h_k(A_j) \end{cases}. \quad (4)$$

with each h_k an independent minwise hash function (See [6]). The estimator is fully efficient, i.e., the variance is given by the Cramér-Rao lower bound

$$\text{var}(\hat{J}) = J(1-J)/K, \quad (5)$$

where we have dropped the Jaccard arguments for brevity. Equation 5 shows that Jaccard coefficients can be approximated to arbitrary precision using minhash signatures with an estimation error that scales as $O(1/\sqrt{K})$.

The memory requirement of minhash signatures is Kn integers, and so can be configured to fit into memory and for $K = 1000$ and $n = 675,000$ is only $\approx 4GB$. In comparison to calculating Jaccard similarities of the largest 675,000

Twitter accounts with $\approx 4 \times 10^{10}$ neighbours minhashing reduces expected processing times by a factor of $10,000$ and storage space by a factor of 1000.

Locality Sensitive Hashing (LSH)

Minhashing dramatically improves comparison times between two accounts, but for large numbers of accounts, finding near-neighbours is expensive. Locality Sensitive Hashing (LSH) is an efficient system for finding approximate near neighbours. LSH has an elegant formulation when combined with minhashing for queries near neighbours in Jaccard space. The minhash signatures are divided into bands containing fixed numbers of hash values and LSH exploits that similar signatures are likely to have identical bands. An LSH table can be constructed that points from each account to all accounts that have at least one identical band. We apply LSH to every input seed independently to find all candidates that are 'near' to at least one seed. In our implementation, we use 500 bands, each containing two hashes. As most accounts share no neighbours, the LSH step dramatically reduces the number of candidate accounts and the algorithm runtime by a factor of roughly 100. Without LSH our algorithm would not run in real-time.

Sorting Similarities

LSH produces a set of candidate accounts that are related to one or more of the input seeds. In general we do not want every candidate returned by LSH and so we must select the subset that are most associated with the whole seed set. We experimented with two sequential ranking schemes: Minhash Similarity (MS) and Agglomerative Clustering (AC). The rankings can best be understood through the Jaccard distance $D = 1 - J$, which is used to define the centre $\mathbf{X} \in [0,1]^M$ of any set of M vertices. At each step AC and MS augment the results set C with A^* the closest account to $\mathbf{X} : A^* \notin C$. However MS uses a constant value of $\mathbf{X}$ based on the input seeds while AC updates $\mathbf{X}$ after each step. Formally, the centre of the input vertices used for MS is defined by

$$X_j(A_j, S) = \frac{1}{n} \sum\nolimits_{i=1}^{n} D(A_j, S_i), \qquad j = 0, 1, ..., M. \quad (6)$$

At each iteration C and $\mathbf{X}$ are updated by first setting $C = S$ and then adding the closest account given by

$$A^* = \arg\min_i X(A_i, C) \quad \forall A_i \notin C \quad (7)$$

leading to

$$C_{t+1} = C_t \cup A^*.$$

The new centre X_{n+1} is most efficiently calculated using the recursive online update equation

$$X_{t+1}(A, C_{t+1}) = \frac{nX(A, C_t) + D(A^*, C_t)}{n+1}. \quad (8)$$

where n is the size of C_t.

4.1.3 Stopping Criterion

Both AC and MS are sequential processes and will return every candidate account unless stopped. The simplest criteria is to stop after a fixed number of iterations. One application of our work is to help define optimal endorsement strategies. In this context we want to answer questions like: "What is the smallest set of closely related athletes that have influence on over half of the users of Twitter?". We refer to the number of unique neighbours of a set of accounts as the *coverage*. An exact solution to this problem is combinatorial. However it can be efficiently approximated using minhash signatures. We exploit two properties of minhash signatures to do this: The unbiased Jaccard estimate through Equation 2 and the minhash signature of the union of two sets is the elementwise minimum of their respective minhash signatures. Minhash signatures allow coverage to be used as a stopping criteria to rank LSH candidates in real-time.

Efficient Coverage Computation.

The coverage y is given by

$$y = \left| \bigcup\nolimits_{i=1}^{n} N(A_i) \right|, \quad (9)$$

the number of unique neighbours of the output vertices. Every time a new account A is added we need to calculate $|N(C) \cup N(A)|$ to update the coverage. This is a large union operation and expensive to perform on each addition. Lemma 1 allows us to rephrase this expensive computation equivalently by using the Jaccard coefficient (available cheaply via the minhash signatures), which we subsequently use for a real-time iterative algorithm.

LEMMA 1. *For a community $C = \bigcup_i A_i$ and a new account $A \notin C$, the number of Neighbours of the union $A^* \cup C$ is given as*

$$|N(A \cup C)| = \frac{|N(A)| + |N(C)|}{1 + J(A, C)}. \quad (10)$$

PROOF. Following (1), the Jaccard coefficient of a new Account $A \notin C$ and the community C is

$$J(A, C) = \frac{|A \cap C|}{|A \cup C|}. \quad (11)$$

By considering the Venn diagram and utilising the inclusion-exclusion principle, we obtain

$$|A \cup C| = |A| + |C| - |A \cap C|. \quad (12)$$

Substituting this expression in the denominator of the Jaccard coefficient in (11) yields

$$\frac{|A| + |C|}{1 + J(A, C)} \overset{(11)}{\overset{(12)}{=}} \frac{|A| + |C|}{1 + \frac{|A \cap C|}{|A| + |C| - |A \cap C|}} = \frac{|A| + |C|}{\frac{|A| + |C|}{|A| + |C| - |A \cap C|}}$$
$$= |A| + |C| - |A \cap C| \overset{(12)}{=} |A \cup C|,$$

which proves (10) and the Lemma. □

Once A^* is determined according to (7), we use Lemma 1 to update the unique neighbour count as

$$|N(C_{t+1})| = \frac{|N(C_t)| + |N(A)|}{1 + J(C_t, A)}. \quad (13)$$

The right hand side of (13) contains three terms: $|N(C_t)|$ is what we started with, $|N(A)|$ is the neighbour count of A, which is easily obtained from Twitter or Facebook metadata and $J(C_t, A)$ is a Jaccard calculation that can rapidly be approximated with minhash signatures[3]. Using this rela-

[3]using the property that the signature of the union of two signatures is their elementwise minimum $sig(\bigcup_i A_i)_j = \min_i(sig(A_i)_j)$

tionship we are able to calculate the coverage with negligible additional calculations.

Once A^* is determined according to (7), we use Lemma 1 to update the unique neighbour count as

$$|C_{t+1}| = \frac{|C_t| + |A^*|}{1 + J(C_t, A^*)},$$

where $|A^*|$ is the neighbour count of A^*, which is easily obtained from Twitter.

Using this relationship we are able to terminate a seed expansion process based on coverage without the stopping criteria calculation dominating the processing time.

4.2 Stage 2: Community Detection and Visualisation

Stage one expanded the seed accounts to find the related region. This was done by first finding a large group of candidates using LSH that were related to any one of the seeds and then filtering down to the accounts most associated to the whole seed set.

In Stage two, the vertices returned by Stage one are used to construct a weighted Jaccard similarity graph. Edge weights are calculated for all pairwise associations from the minhash signatures through Equation 2. This process effectively embeds the original graph in a metric Jaccard space [5]. Community detection is run on the weighted graph.

The final element of the process is to visualise the community structure and association strengths in the region of the input seeds. We experimented with several global community detection algorithms. These included INFOMAP, Label Propagation, various spectral methods and Modularity Maximisation [22, 21, 19]. The Jaccard similarity graph is weighted and almost fully connected and most community detection algorithms are designed for binary sparse graphs. As a result, all methods with the exception of label propagation and WALKTRAP were too slow for our use case. Label Propagation had a tendency to select a single giant cluster, thus adding no useful information. Thus, we chose WALKTRAP for community visualisation.

5. EXPERIMENTAL EVALUATION

We assess the affect of LSH and minhash approximations and demonstrate the quality of our results in three experiments: (1) We measure the sensitivity of the Jaccard similarity estimates with respect to the number of hash functions used to generate the signatures. This will justify the use of the minhash approximation for computing approximate Jaccard similarities. (2) We compare the run time and recall of our process on ground-truth communities with PPR (state of the art) on a single laptop. (3) We visualise detected communities and demonstrate that association maps for social networks using minhashing and LSH produce intuitively interpretable maps of the Twitter and Facebook graphs in real-time on a single machine.

5.1 Experiment 1: Assessing the Quality of Jaccard Estimates

We empirically evaluate the minhash Jaccard estimation error by comparing estimates (Equation 2) to exact values(Equation 1) using a sample of 400,000 pairwise relationships from the 675 thousand hashed accounts. Figure 1 shows the estimation error (L1 norm) as a function of the signature length. Standard error bars are just visible up until 400 hashes. The graph shows an expected error in the Jaccard of just 0.001 at 1,000 hashes. Due to the high degree of accuracy and diminishing improvements we select a signature length of $K = 1,000$, which provides an appropriate balance between accuracy and performance (both runtime and memory scale linearly with K).

Table 3: Most similar accounts to @Nike. J and R give the true Jaccard coefficient and Rank, respectively. $\hat{J}$ and $\hat{R}$ give approximations using Equation (2) where the superscript determines the number of hashes used. Signatures of length 1,000 largely recover the true Rank.

Twitter handle	J	R	$\hat{J}^{100}$	$\hat{R}^{100}$	$\hat{J}^{1000}$	$\hat{R}^{1000}$
adidas	0.261	1	0.22	2	0.265	1
nikestore	0.246	2	0.25	1	0.255	2
adidasoriginals	0.200	3	0.18	3	0.222	3
Jumpman23	0.172	4	0.13	7	0.166	4
nikesportswear	0.147	5	0.18	4	0.137	5
nikebasketball	0.144	6	0.16	5	0.127	7
PUMA	0.132	7	0.13	6	0.132	6
nikefootball	0.127	8	0.08	17	0.110	9
adidasfootball	0.112	9	0.09	16	0.113	8
footlocker	0.096	10	0.08	17	0.096	11

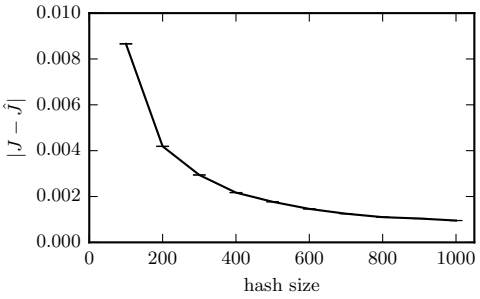


Figure 1: Expected error from Jaccard estimation using minhash singatures against the number of the hashes used in the signature. The error bars show twice the standard error using 400,000 data points.

A typical top ten list of Jaccard similarities is given in Table 3 for Nike (based on the true Jaccard). Possible matches include sports people, musicians, actors, politicians, educational institutions, media platforms and businesses from all sectors of the economy. Of these, our approach identified four of Nike's biggest competitors, five Nike sub-brands and a major retailer of Nike products as the most associated accounts. This is consistent with our assertion that the Jaccard similarity of neighbourhood sets provides a robust similarity measure between accounts. We found similar trends throughout the data and this is consistent with the experience of analysts at Starcount using the tool. Table 3 also shows how the size of the minhash signature affects the Jaccard estimate and the corresponding rank of similar accounts. Local community detection algorithms add accounts in similarity order. Therefore, approximating the true ordering is an important property.

5.2 Experiment 2: Comparison of Community Detection with PPR

In experiment 2 we move from assessing a single compo-

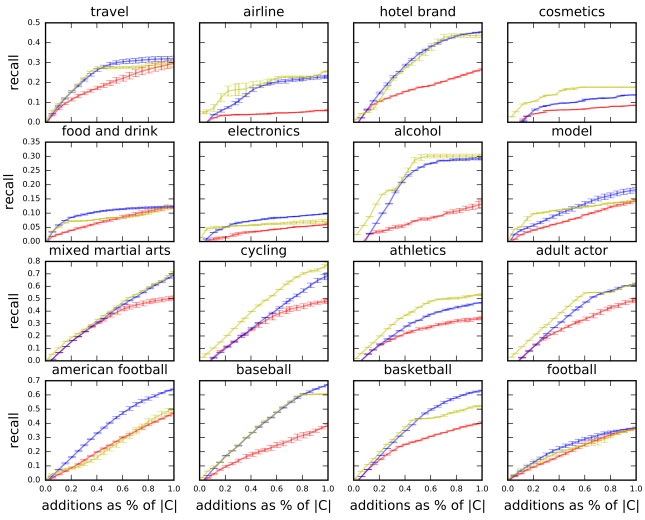


Figure 2: Average recall (with standard errors) of Agglomerative Clustering (yellow), Personal PageRank (red) and Minhash Similarity (blue) against the number of additions to the community expressed as a fraction of the size of the ground-truth communities. The tight error bars indicate that the methods are robust to the choice of seeds.

nent (minhashing) to a system wide evaluation. We evaluate the ability of our algorithm to detect related entities by measuring its performance as a local community detection algorithm seeded with members of the ground-truth communities. To generate ground-truth communities we attempted to match Wikipedia pages to every Twitter account. Where a match was possible we extracted the Wikipedia tag that was synonymous with function and placed accounts with the same tag in the same ground-truth community. As a baseline for comparison we use the (state of the art) Personal Page Rank (PPR) algorithm. It is impossible to provide a fully like-for-like comparison with PPR. Running PPR on the full graph (700 million vertices and 20 billion edges) that we extract features from requires cluster computing and could return results outside of the accounts we hashed. The alternative is to restrict PPR to run on the directly observed network of the largest Twitter accounts, which could then be run on a single machine. We adopt this later approach as it is the only option that meets our user requirements (single machine and real-time).

In our experimentation, we randomly sample 30 seeds from each ground-truth community. To produce MS and AC results the seeds are input to an LSH query, which produces a list of candidate near-neighbours. For each candidate the Jaccard similarity is estimated using minhash signatures and sorted by either the MS or AC procedures. Our PPR implementation uses the 30 seeds as the *teleport* set and runs for three iterations returning a ranked list of similar accounts.

In all cases, we sequentially select accounts in similarity order and measure the recall after each selection.

$$\text{recall} = \frac{|C \cap C_{\text{true}}|}{|C_{\text{true}}| - |C_{\text{init}}|} \tag{14}$$

with C_{init} as the seeds, C_{true} as the ground truth community and C as the set of accounts added to the output. For a community of size $|C|$ we do this for the $|C|-30$ most similar accounts so that a perfect system would pass through $(1, 1)$.

Table 4: Area under the recall curves (Figure 2). Bold entries indicate the best performing method. Minhash similarity (MS) is the best method in 8 cases, Agglomerative Clustering (AC) in 8 cases and Personalised PageRank (PPR) in none. A perfect community detector would score 0.5

tags	size	PPR	MS	AC
travel	2038	0.186	**0.240**	0.230
airline	363	0.040	0.151	**0.180**
hotel brand	836	0.160	**0.294**	0.285
cosmetics	332	0.055	0.086	**0.143**
food and drink	2974	0.072	**0.099**	0.082
electronics	689	0.035	**0.069**	0.059
alcohol	388	0.069	0.199	**0.229**
model	2096	0.078	**0.110**	0.109
mixed martial arts	751	0.317	0.363	**0.386**
cycling	371	0.278	0.330	**0.445**
athletics	530	0.219	0.285	**0.365**
adult actor	352	0.269	0.347	**0.397**
american football	1295	0.240	**0.371**	0.240
baseball	616	0.203	**0.379**	0.378
basketball	786	0.252	**0.380**	0.353
football	4111	0.202	**0.233**	0.212

Table 5: Clustering runtimes averaged over communities.

PPR	MS	AC
12.58 ± 8.83	0.23 ± 0.08	18.6 ± 22.0

The results of this experiment for 16 different ground-truth communities are shown in Figure 2 with the Area Under the Curves (AUC) given in Table 4. Bold entries in Table 4 indicate the best performing method. In all cases MS and AC are superior to PPR. Figure 2 shows standard errors over five randomly chosen input sets of 30 accounts from C_{true}. The confidence bounds are tight indicating that the methods are robust to the choice of input seeds.

Table 5 gives the mean and standard deviation of the run times averaged over the 16 communities. MS is the only real-time method and is fastest by two orders of magnitude.

5.3 Experiment 3: Real-Time Graph Analysis and Visualisation

In experiment 3 we provide example applications of our system to graph analysis. Users need only input a set of seeds, wait a quarter of a second and the system discovers the structure of the graph in the region of the seeds. Users can then iterate the input seeds based on what previous outputs reveal about the graph structure. Figure 3a shows results on the Facebook Page Engagements network while Figure 3b uses the Twitter Followers graph. Seeds are passed to the MS process, which returns the 100 most related entities. All pairwise Jaccard estimates are then calculated using the minhash signatures and the resulting weighted adjacency matrix is passed to the WALKTRAP global community detection algorithm [20]. The result is a weighted graph with community affiliations for each vertex. In our visualisations we use the Force Atlas 2 algorithm to lay out the vertices. The thickness of the edges between vertices represents the pairwise Jaccard similarity, which has been thresholded for image clarity. The vertex size represents the weighted degree of the vertex, but is logarithmically scaled to be between 1 and 50 pixels. The vertex colours depict the different communities found by the WALKTRAP community detection algorithm.

Our work uses the richness of social media data to provide insights into a broad range of questions. Two examples are:

- **Describe the factions and relationships within the US Republican party?** This is a question with a major temporal component, and so we use the Facebook Pages graph. We feed "Donald Trump", "Marco Rubio", "Ted Cruz", "Ben Carson" and "Jeb Bush" as seeds into the system and wait for 0.25 s for Figure 3a, which shows a densely connected core group of active politicians with Donald Trump at the periphery surrounded by a largely disconnected set of right-wing interest bodies.
- **How are the major social networks used?** We feed the seeds "Twitter", "Facebook", "YouTube" and "Instagram" into the system loaded with the Twitter Followers graph and wait for 0.25 s for Figure 3b, which shows that Google is highly associated with other technology brands while Instagram is closely related to celebrity and YouTube and Facebook are linked to sports and politics.

In all cases, the user selects a group of seeds (or a single seed) and runs the system, which returns a Figure and a table of community memberships in 0.25 s. Analysts can then use the results to supplement the seed list with new entities or use the table of community members from a single WALKTRAP sub-community to explore higher resolution.

6. CONCLUSION AND FUTURE WORK

We have presented a real-time system to analyse large social networks on a laptop. The key idea is to compress the graph into memory using minhash signatures, which can be used to query Jaccard similarity in real-time through Locality Sensitive Hashing. We have shown that the quality of results achieved significantly outperforms the state of the art operating under the same constraints. Our work has clear applications for knowledge discovery processes that currently rely upon slow and expensive manual procedures, such as focus groups and telephone polling. In general, it offers a potential for organisations to rapidly acquire knowledge of new territories and supplies an alternative monetisation scheme for data owners.

7. REFERENCES

[1] M. Adler and M. Mitzenmacher. Towards compressing web graphs. *DCC*, 2001.

[2] A. Andoni, P. Indyk, H. L. Nguyen, and I. Razenshteyn. Beyond Locality-Sensitive Hashing. In *SODA*, 2014.

[3] V. D. Blondel, J.-l. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of community hierarchies in large networks. In *Journal of statistical mechanics: theory and experiment*, 10008, 2008.

[4] P. Boldi and S. Vigna. The webgraph framework I: compression techniques. In *WWWW*, 2004.

[5] A. Broder. On the resemblance and containment of documents. In *SEQUENCES*, 1997

[6] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[7] F. Chierichetti, R. Kumar, and S. Lattanzi. On Compressing Social Networks. In *SIGKDD*, 2009.

[8] S. Fortunato. Community detection in graphs. In *Physics Reports*, 486(3):75–174, 2010.

[9] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh. WTF: The Who to Follow Service at Twitter. In *WWW*, 2013.

[10] T. Haveliwala, A. Gionis, and P. Indyk. Scalable techniques for clustering the web. In *WebDB*, 2000.

[11] T. H. Haveliwala. Topic-sensitive PageRank. In *WWW*, 2002.

[12] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, 1998.

[13] I. Kloumann and J. Kleinberg. Community membership identification from small seed sets. In *SIGKDD*, 2014.

[14] A. Kyrola, G. Blelloch, and C. Guestrin. Graphchi: Large-scale graph computation on just a pc. In *USENIX*, 2012.

[15] P. Li and C. König. b-Bit minwise hashing. In *WWW*, 2009.

[16] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a Feather: Homophily in Social Networks. In *Annual Review of Sociology*, 27(1):415–444, 8 2001.

[17] M. Mitzenmacher, R. Pagh, and N. Pham. Efficient estimation for high similarities using odd sketches. In *WWW*, 2014.

[18] M. Newman. Fast algorithm for detecting community structure in networks. In *Physical review E*, 69(6), 066133, 2004.

[19] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. In *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 74(3):1-19, 2006.

[20] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS* 284-293, 2005.

[21] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. In *Physical Review E*, 76(3):036106, 9 2007.

[22] M. Rosvall and C. Bergstrom. Maps of random walks on complex networks reveal community structure. In *Proceedings of the National Academy of Sciences*, 105.4: 1118-1123, 2008.

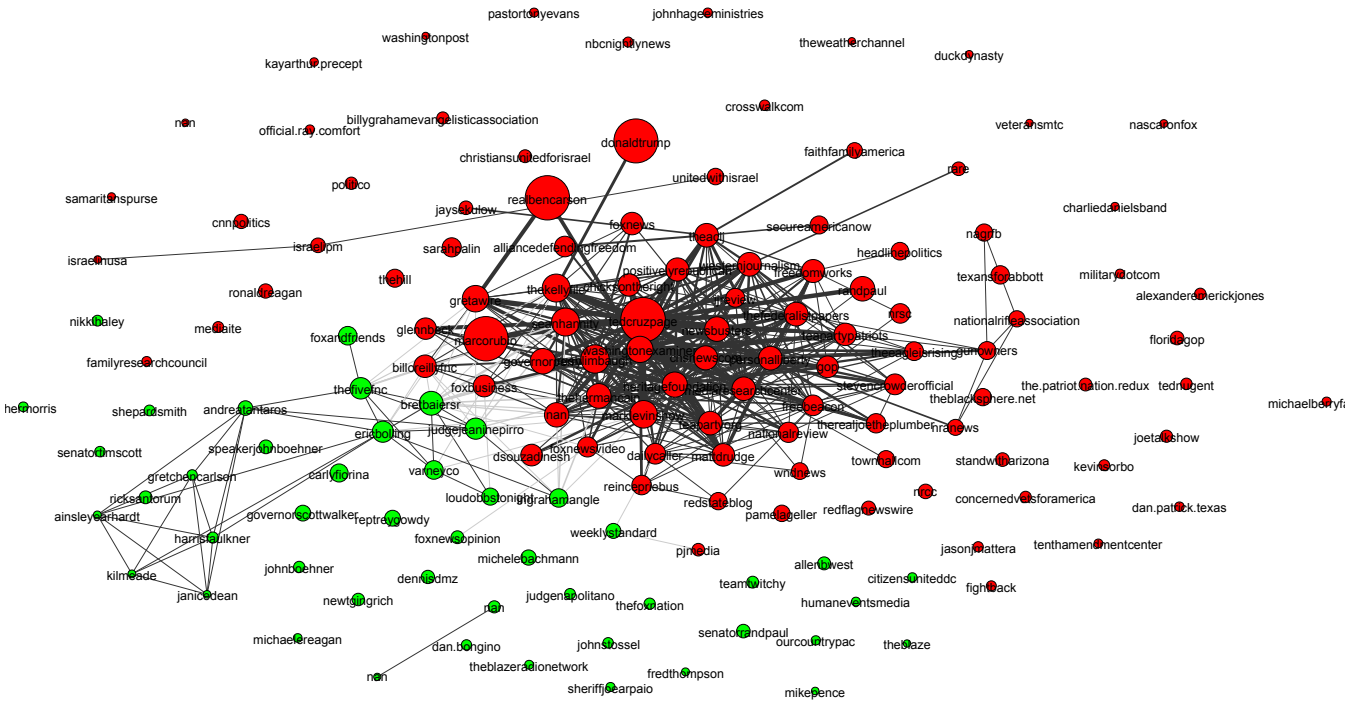
[23] S. E. Schaeffer. Graph clustering. In *Computer Science Review*, 1(1):27âĂŞ64, 2007.

[24] D. F. Gleich and C. Seshadhri. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *SIGKDD*, 2012.

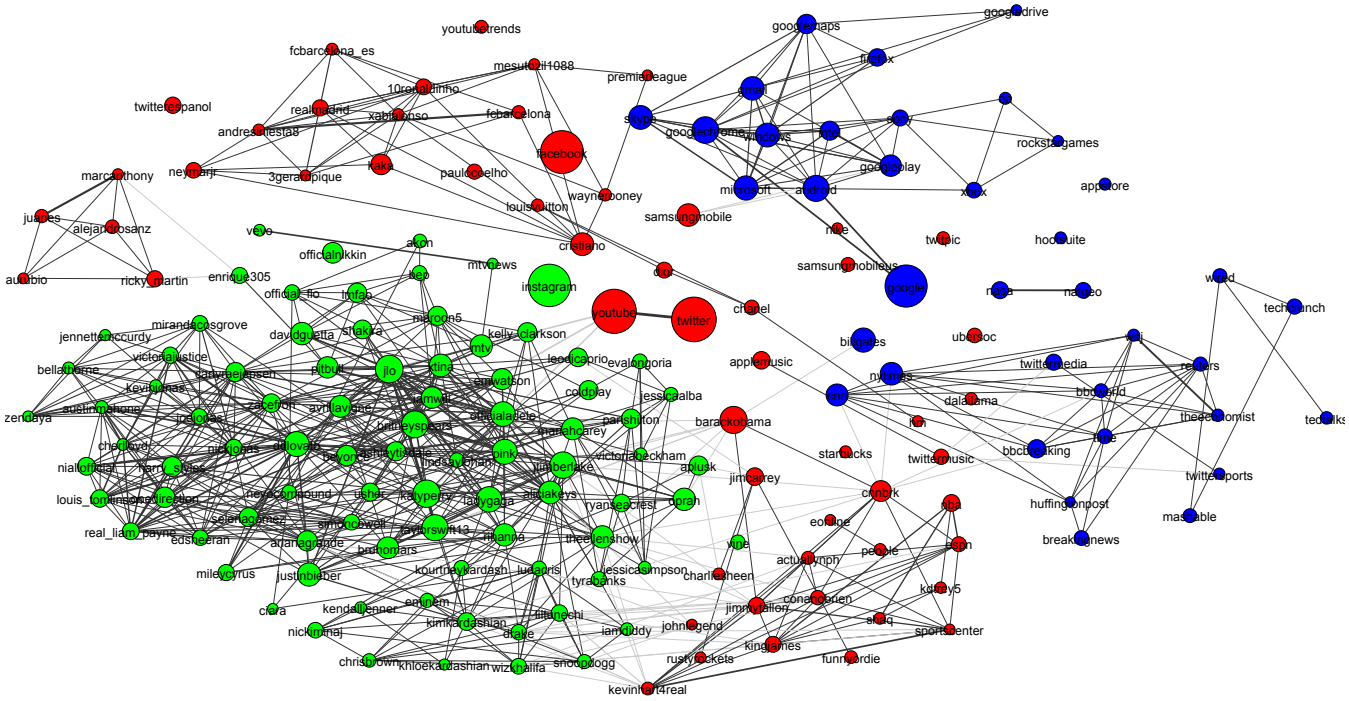
[25] Y. Li, K. He, D. Bindel and J. E. Hopcroft. Uncovering the small community structure in large networks: A local spectral approach. In *WWW*, 2015.

[26] R. Wysocki and W. Zabierowski. Twisted framework on game server example. In *CADSM*, 2011.

[27] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*, 2013.



(a) The US republican party Facebook Page likes graph. Seeds are “Donald Trump”, “Marco Rubio”, “Ted Cruz”, “Ben Carson” and “Jeb Bush”



(b) The major social networks Twitter Follower graph. Seeds are Twitter, Facebook, YouTube and Instagram.

Figure 3: Visualisations of results using different sets of seeds. The vertex size depicts degree of similarity to the seeds. Edge widths show pairwise similarities. Colours are used to show different communities.