Extracting Topics with Focused Communities in Multi-dimensional Social Data

Theodore Georgiou  
Department of Computer Science  
University of California, Santa Barbara  
teo.georgiou@cs.ucsb.edu

Amr El Abbadi  
Department of Computer Science  
University of California, Santa Barbara  
amr@cs.ucsb.edu

Xifeng Yan  
Department of Computer Science  
University of California, Santa Barbara  
xyan@cs.ucsb.edu

ABSTRACT
Understanding what social media users discuss and what is happening in the real world can be enabled through the automatic analysis and summarization of Online Social Media. Trend Discovery, through the extraction of trending topics, is utilized in marketing campaigns and by companies to identify customer interests and potential new markets. While there is a plethora of techniques to identify trending topics, there is a lack of focus on the characteristics of the underlying population that participate in a trend. Users that mention a topic define a multivariate vector of demographics and user characteristics with the potential to offer insights into the communities that are focused on a topic, both latent and obvious. We propose a novel algorithmic framework for the efficient and scalable extraction of the combination of user characteristics that define a community interested in a topic. Potential results include cases like “female residents of Boston, MA, that support the Democratic party, are focused on the activist topic #FreeJustina” or “young adults, living in the US, are focused on topic #NavyYardShooting with a Negative sentiment”. Such topics might be significantly popular or not, but community extraction emphasizes the importance and focused interest in a topic even if it is not as popular as other topics with a more defused audience. The proposed framework can support any number of attributes and scales linearly. We assessed our algorithm’s accuracy and efficiency on synthetic and real Twitter data and results show high accuracy and efficiency.

1. INTRODUCTION
The study of social patterns in Online Social Media like Twitter or Facebook can be very helpful in identifying collective user behavior among specific segments of society. Trending Topics have been popularly used in the detection of breaking news, as well as in marketing and advertising mechanisms. In its general definition, a Trending Topic is a collection of words or phrases that refer to a temporarily popular topic. Usually, the origin of a trending topic is a popular real life event that is being discussed on social media or a meme that is spreading. Trending topics are used to understand and explain how information and memes diffuse through vast social networks with hundreds of millions of nodes. Many algorithms have been proposed for discovering interesting trending topics utilizing techniques from the areas of Anomaly Detection, Data Streams, and Clustering. In existing studies, trending topics are mined for specific interest areas like Sports [18], Earthquakes [21], News reporting [16, 22], or general event detection [20, 1]. In this paper we research the novel idea of identifying the underlying user communities that are interested in social media topics and then utilize this knowledge to rank topics by importance. Such a task can be challenging in terms of complexity when dealing with a non trivial number of community characteristics. The official Twitter Trending Topics are personalized to the user by displaying the top topics from categories the user is interested in. This is a simple approach to serve relevant trends but focuses only on topic categories (e.g. Technology, Politics, etc.) or location, and does not identify topics where the underlying population has specific properties, thus, can miss less popular topics with highly interesting community characteristics.

We observed that the user population involved in a trend offers high potential in understanding a trend and how other users might react to it. In a previous study on Twitter trending topics even simple social relations between the participants of a topic trend could greatly enhance the understanding of trending topics [9] or spammer detection [7]. Then, a space efficient framework was proposed, named GeoScope [10], that extracts trending topics which are highly focused in specific geographical locations. Human evaluations showed that topics with a high geographical correlation tend to be more interesting than topics with a dispersed population.

In this paper, we propose a novel community detection algorithm utilizing a spectrum of social characteristics rather than just geographic locations. The detection of community characteristics that are interested in a topic, like gender, age, location, race, ethnicity, political affiliation, etc., can yield powerful results which are useful in a variety of domains. Marketers can understand their customers better by identifying the communities interested in their or competitive products. Advertisements, which usually are linked to a trending topic or event, can become more personalized. Content recommendation can be improved through the extraction of target groups interested in specific topics. Last but not least, the understanding of the underlying population of a topic can enable the study and discovery of latent communities that sociologists might not be aware of.

Communities focused on topics, can sometimes be expected and sometimes unexpected. It is easy to anticipate that young boys will be interested in the PlayStation 4 gaming console even without monitoring the widely popular trending topic #PS4. But we might not expect that women in the area of Boston, MA, that also support the Democratic party, showed their solidarity to an arrested female
teen named Justina with the not so popular topic #FreeJustina. It is
even more unexpected to observe the hijacking of the hashtag
campaign #ReasonsToVisitEgypt that was originally created to pro-
mote tourism in Egypt, but local citizens used it negatively to raise
awareness for the country’s political situation. The important take
away is that using only the popularity of a trending topic is not
enough; a better understanding of the underlying community can
yield a better ranking for interesting topics that might not be glob-
ally popular.

We define a specific type of community that we call focused com-
munity to enable the efficient extraction of communities interested
in topics on social media. We exploit specific properties of this defi-
nition to propose a novel framework that receives a social stream as
its input and reports topics and the corresponding focused commu-
nities as its output. The framework scales linearly with the number
of attributes, and reports communities that might lie within any
attribute combination or sub-dimension.

The current research project makes the following contributions:

- The definition of a maximally focused community.
- Information Retrieval metrics, like TF/IDF, are adapted to rank
topics based on their communities’ relative popularity and excludiv-
y
- A probabilistic and scalable algorithm for the discovery of
maximally focused communities with amortized linear time
complexity. The algorithm also deals with missing values in
records, in a stochastic way that maintains high confidence
on the reported results.

- The effectiveness and efficiency of the proposed approach is
tested through extensive experimentation and analysis on a
massive social dataset from Twitter and synthetic data.

Related work is presented in Section 2. The definition of a Fo-
cused Community is given in Section 3 and the description of
the proposed algorithm is offered in Section 4. The methodology for
dealing with missing values in real social data is described in
Section 5 and, finally, experimental results are presented in Section 6.

2. RELATED WORK

While there are multiple approaches to extract trending topics
from social content we are the first to explore the underlying fo-
cused communities. Our work builds on many techniques in areas
that share common properties with this problem, most notably from
Subspace Clustering and Frequent Itemset Extraction/Association
Rule Mining.

Association Rule Mining using Frequent Itemset Extraction [6]
is a well studied area and poses similarities to the attribute-based
community extraction. Techniques that sample the data to perform
fast itemset extraction are the closest to our proposed approach
since probabilistic algorithms are used to reduce complexity. Such
techniques include Toivonen [25] and Chakaravarthy et al. [11].

Clustering algorithms for data in multiple dimensions, known as
subspace clustering algorithms, are usually divided in two cat-
egories: density-based methods and k-means-based methods. A
detailed survey on both categories can be found in [15]. Density-
based methods assume that a cluster is a data space region in which
the element distribution is dense and is separated from other clus-
ters by regions of low density. Examples of such algorithms include
CLIQUE [5], P3C [17], and 4C [8]. K-means based algorithms
come closer to the nature of our framework for community extrac-
tion of a trending topic. In the k-means and the k-medoids [14] meth-
ods k cluster centroids are initially picked and through an iterative
process, where each datapoint is assigned to its closest cluster cen-
troid, convergence is eventually achieved – or a maximum number
of iterations is reached. Subspace clustering methods that follow
this approach include PROCLUS [3], and ORCLUS [4]. The same
algorithmic principles are also used to solve the frequent itemset
and association rule mining problems (e.g. a-priori pruning in used
in both [6] and CLIQUE [5]).

Our approach is similar to the k-means based methods in the
sense that k datapoints are also randomly picked at the beginning,
although this number, k, does not dictate the final number of the
extracted communities which, in fact, might be less than k. In con-
trast, our method probabilistically locates the sub-space of a com-
munity interested in a Trending Topic starting with these k random
datapoints and then climbs through the attribute lattice to identify
the interested community. Thus, we can detect focused communities
quickly, a task that is possible but inefficient through existing
sub-space clustering and association rule mining techniques.

Missing value imputation (handling) is important given the na-
ture of social data, something that is consistently overseen in tradi-
tional subspace clustering and frequent itemset methods [13]. Typ-
ically, records with missing values are completely removed and al-
gorithms are then applied only on the remaining or a subset of the
records. Our algorithmic framework includes a stochastic method-
ology to deal with missing values in a less severe way and yields
results with high confidence that otherwise could be discarded if
records were completely removed.

Similar to our approach, probabilistic or Monte Carlo based meth-
ods for community extraction have also been explored in Perozzi
et al. [19]. They study extracting community attributes that form
highly connected subgraphs within the social network. To detect
the correct values for each attribute they utilize Monte Carlo sam-
pling to randomly select values until a connected subgraph is formed.
Our approach seeds the process by sampling k datapoints from a
trending topic’s population. This leads to a much more agile and
efficient attribute value selection process.

3. FOCUSED COMMUNITIES

The process of identifying focused communities has to be ap-
plied on each topic individually. To extract and understand the un-
derlying communities interested in a particular topic, T, two pieces
of information are necessary. 1) The topic population P which in-
cludes every social posting that mentions topic T. We will refer
to these social postings using the general term datapoints but in the
specific case of Twitter they are called tweets. 2) The corre-
sponding social characteristics (attribute values) for every data-
point. These attributes can include user demographics like Loca-
tion, Age, Gender, Race, or characteristics like political affiliation,
supporting soccer team, hobbies, etc. Each user that mentions a
topic can be represented by an attribute value vector. For example,
a hypothetical 5-dimensional attribute vector could be: [Location:
Los Angeles, Age: 18, Gender: Male, Citizenship: USA, Political
Affiliation: Republican]. Certain attributes can be hierarchical, like
Location or Age. If a user lives in Los Angeles, then she also lives
in California, or USA, or the World. If a user is 15 years old then
she also belongs in the “teenager” age group. Ultimately, given the
population of a topic T, we want to extract a value for each attribute
in order to discover a “maximally focused community” interested
in topic T. In the current section we will formally define what a
focused and maximally focused community is.

Let D be a domain with N attributes. Each attribute a_i ∈ D has
a finite set of values V_{a_i}. Let P be a set of datapoints where each
datapoint is represented by a vector of N attribute values v_i ∈ V_{a_i}.
We refer to any vector of attribute values as a tuple; therefore, any
A datapoint is considered a tuple. The support of a single attribute value is equal to the number of datapoints in $P$ that contain the value. The support of a tuple instance is equal to the number of datapoints with values that match the values of the tuple.

Categorical attribute values may follow a tree-like hierarchical pattern. The Location attribute can be described using a tree hierarchy of 4 levels: city, region/state/province, country, and “Worldwide”. The Ethnicity attribute can be described using a tree hierarchy of 3 levels: ethnicity, minority/non minority, and “Any ethnicity”. Values in each level of the hierarchy are connected to a single ancestor from the previous level and to an arbitrary number of successors in the next level (which can be zero for the values of the bottom level). The root of the hierarchy is symbolized with the value ‘*’. Note that every attribute can be described at least by the trivial hierarchy of 2 levels where the bottom level contains all the values and the top level contains just ‘*’. Numerical attribute values can be viewed as hierarchical attributes as well. Using a radius $r$ the hierarchical ancestor of a numerical value $v$ can be dynamically estimated as the range $[v-r, v+r]$. Alternatively, the values of a numerical attribute can be discretized so it becomes categorical. In the current work we focused on categorical attributes but the proposed algorithm works also with numerical attributes.

We define a homophilic community $C$ to be a partition of the population $P$ for a specific topic $T$ which share common values for a subset of the $N$ attributes. For example, a homophilic community can include all the users who live in Los Angeles and are male. Homophilic communities are important because homophily dictates that every member of the community interested in $T$ share some common characteristics. However, the community members are not necessarily connected in the social graph. From this point on we will refer to homophilic communities simply as communities.

For a given topic with a group of datapoints $P$, a community $C \subseteq P$ is defined by $N$ attribute values $\{v_1 \in a_1, v_2 \in a_2, \ldots, v_N \in a_N\}$ and can also be represented as a tuple. We will use the symbol $C$ to refer to the members of the community and $C_v$ to refer to the tuple with the attribute values that define the community. We define a community $C$ to be focused if there is at least one attribute value $v \in C_v$ which no datapoint in $P - C$ shares. This attribute value $v$ is an exclusive feature of the community: every member of the community has this value because of the homophily property but no other person in $P$ shares it. Let $P_v$ be the set of all datapoints in $P$ that contain the attribute value $v$. The following must hold for $C$ to be focused: $\exists v \in C_v$ so that $P_v \equiv C$.

Figure 1 illustrates the difference between an arbitrary community (non focused) and a focused community with three attributes. As an example we can assume that attribute $a$ is Location with value $v_a$ equal to Los Angeles, attribute $b$ is Age with value $v_b$ equal to 18 years old, and attribute $c$ is Gender with value $v_c$ equal to Male. In the first case, the population corresponding to the intersection of the three attributes defines a non focused community, i.e., 18 year old males who live in Los Angeles. In the second case, the population corresponding to the attribute $v_c \equiv 18$ years old is almost identical to the intersection of all three attributes. Therefore, the support of the Los Angeles male community is almost equal to the number of users in $P$ that are 18 years old, since almost none else in $P - C$ has this age.

We can relax the definition of a focused community by introducing a relaxation threshold $\epsilon$ so that we can measure how close a community is to being perfectly focused. The support of a community $C$ must be smaller or equal to the size of $P_v$: $|C| \leq |P_v|$. If $P_v \neq \emptyset$ this inequality can be re-written as:

$$\frac{|C|}{|P_v|} \leq 1 \Rightarrow \frac{|C|}{|P_v|} - 1 \leq 0$$

We can relax the definition of a focused community by introducing a relaxation threshold $\epsilon$ so that we can measure how close a community is to being perfectly focused. The support of a community $C$ must be smaller or equal to the size of $P_v$: $|C| \leq |P_v|$. If $P_v \neq \emptyset$ this inequality can be re-written as:

$$\frac{|C|}{|P_v|} - 1 \leq \epsilon$$

We call the absolute value of the fraction at the left side of the inequality the focus metric of the community. A value of 0 indicates that the community is perfectly focused. A value above $\epsilon$ indicates that the community is not focused.

The relaxation of the focused community definition is necessary for the discovery of focused communities in the presence of data noise or missing values. The threshold $\epsilon$ practically defines how many datapoints in $P$ might share the “exclusive” feature of the focused community without being part of it.

The generalization of an attribute value $v$ is the process of replacing the value $v$ with a direct ancestor of $v$ in the attribute’s hierarchy. The generalization of any value except ‘*’ is possible. The root value ‘*’ cannot be generalized since it has no ancestors. We denote the case of a missing attribute value using the “⊥” operator (bottom). A “⊥” value can be directly generalized to ‘*’ through a single generalization step no matter how high the attribute hierarchy is. In the general case, an attribute $a$ can be generalized from value $v_a$ to value $v_b$ if $v_b$ precedes or is equal to $v_a$, in attribute $a$’s hierarchy. We denote this relation between $v_a$ and $v_b$ using the operators $\geq$ (succeeds) and $\leq$ (precedes): $v_a \geq v_b$ or $v_a \leq v_b$.

As an example, for the Location attribute the following relations are true: Los Angeles $\geq$ Los Angeles, Los Angeles $\geq$ California, Los Angeles $\geq$ USA, California $\geq$ USA, California $\geq$ *.

The support of a generalized attribute value in $P$ is equal to the number of datapoints that contain any successor of the value. For
4. Algorithm Overview

Focused communities interested in each topic. The algorithm is the ranked list of topics and the extracted maximally focused community to meet a minimum support requirement, relative to the population. More specifically, we introduce a support threshold $\xi \leq 1$ so that every maximally focused community has support of at least $\xi|P|$.

4. ALGORITHM DESCRIPTION

The proposed algorithm aims to extract a new topic ranking that utilizes the identified focused communities. Given a set of topics, all the maximally focused communities of each topic with a focus metric less or equal to $\epsilon$ and support greater or equal to $\xi$ are extracted. The community characteristics are then used to calculate a score which leads to a novel topic ranking. The output of the algorithm is the ranked list of topics and the extracted maximally focused communities interested in each topic.

4.1 Algorithm Overview

Figure 3 illustrates the overview of the algorithm. Given a stream of datapoints being traversed through a sliding window, topics are identified (keywords, entities, phrases, hashtags) and the datapoints of the window are grouped by topic. All the topics with at least a minimum number of datapoints form the pool of candidate topics. For each topic in the candidate topics, the focused communities are extracted through the Sample&Climb module (dashed rectangle in Figure 3). The extraction of a maximally focused community is an optimization problem: find a combination of attribute values (tuple $C_t$) that maximizes the size of the community defined by $C_t$, while minimizing the focus metric (Equation (2) from Section 3). The Sample & Climb algorithm selects a random sample of datapoints from $P$ (seeding phase) and uses each datapoint as a starting point to reach the attribute values of a focused community through a series of value generalizations (climbing phase). The output of the Sample&Climb module is used to calculate a topic score for each topic. These scores are then used to rank the pool of candidate topics of the sliding window.

4.2 Sample&Climb Seeding Phase

The Seeding phase must efficiently bootstrap the optimization problem of extracting a tuple that defines a topic’s maximally focused community. To that end, we uniformly sample $k$ tuples from $P$ (datapoints) and create a new set $S$: every tuple $t \in S$ is then fed to the climbing phase which will reach a potential maximally focused community. If the sampled tuple is actually a member of a maximally focused community (checked by Equation 2), the climbing phase should extract the community. If the sampled tuple is not a member of any focused community the climbing phase will not extract a community. The intuition behind this approach is to probabilistically select datapoints that might belong to a maximally focused community. This intuition is visualized in Figure 4 where we assume that in a population $P$, a focused community $C_1$ and $C_2$ exist. The sampling of datapoints $d_1$ or $d_2$ can enable the extraction of community $C_1$. The sampling of datapoint $d_4$ can enable the extraction of community $C_2$. The sampling of datapoint $d_3$ does not enable the extraction of any community and a different datapoint needs to be sampled. If the datapoint is indeed a member of a community, then a series of attribute generalizations and focus metric computations can lead us to the actual attribute values of the community. For example, if the following focused community exists: [Location: USA, Gender: *, Age: 13-18] and the datapoint: [Santa Barbara, Male, 18] is randomly selected then the location value can be generalized twice (Santa Barbara $\rightarrow$ California $\rightarrow$ USA),
the gender value once (Male → *), and the age value once (18 → 18-23) to reach the tuple that describes the community.

When a sampled tuple successfully leads to the extraction of a maximally focused community, the result is saved. If the next sampled tuple succeeds an already extracted community, by a previous iteration in the seeding phase, then the tuple is skipped since it can only lead to a known community and would be a waste of resources to process it. Pseudocode for the seeding phase is provided in Algorithm 1. Line 5 tests if the new sampled tuple is succeeds an already extracted community. If the tuple is already a successor of an extracted community, the climbing phase is skipped since it will yield the same result given that the climbing process is deterministic. The returned result of the climbing phase is a maximally focused community if climbing was successful, or equal to NULL if a focused community could not be extracted (line 8).

Algorithm 1: Seeding phase

\[
\begin{align*}
\textbf{Data:} & \ Tuples P, \ attribute \ hierarchies H[N] \\
\textbf{Result:} & \ Set \ of \ maximally \ focused \ communities C \\
1 & \text{begin} \\
2 & \text{C} \leftarrow \{\}; \\
3 & \text{S} \leftarrow \text{sample}(P); \\
4 & \text{for} \ t \in S \text{ do} \\
5 & \quad // \ If \ t \ succeeds \ any \ community \ c \ in \ C, \\
6 & \quad // \ skip \ it \ (optimization). \\
7 & \quad \text{if} \ \exists c \in C \ t \geq c \text{ then} \\
8 & \quad \quad \text{continue}; \\
9 & \quad c \leftarrow \text{climb}(t,P,H); \\
10 & \quad // \ c \ is \ a \ community \ tuple \\
11 & \quad \text{if} \ c \neq \text{NULL} \text{ then} \\
12 & \quad \quad C \leftarrow C \cup \{c\}; \\
13 & \text{end}
\end{align*}
\]

Based on the desired success probability of the seeding phase \( p_b \), the appropriate minimum size of the sample \( S \) can be determined. Let \( k \) be the number of datapoints sampled during the Seeding phase and \( C \) a unique maximally focused community in \( P \).

The sampling of datapoints can be simulated through a series of Bernoulli trials where success is defined as the selection of a datapoint tuple \( t \) so that \( t \geq C_t \). The number of trials is equal to the size of the sample: \( |S| = k \). The probability of success in a single trial is equal to \( p = \frac{|C_t|}{|P|} \). The probability of at least one success out of \( k \) trials (we can assume that \( P \) is large enough for the trials to be independent even without replacement) is equal to \( 1 - \frac{k}{k} \) minus the probability of getting 0 successes. This probability is defined by the geometric equation that describes the CDF of \( k \) Bernoulli trials: \( 1 - (1 - p)^k \). Therefore, we have:

\[
\begin{align*}
p_b &= 1 - (1 - |C_t|/|P|)^k = 1 - (1 - p)^k \quad (3)
\end{align*}
\]

We want to find the minimum value of \( k \) so that the right hand of Equation (3) is greater or equal to \( p_b \). Let \( q = 1 - p \) be the probability of failure in a single trial.

\[
\begin{align*}
p_b \leq 1 - (1 - p)^k & \quad \Rightarrow \quad q^k \leq 1 - p_b \quad \Rightarrow \quad q < 1 \\
k \geq \log_q(1 - p_b) & \quad \Rightarrow \quad k \geq \frac{\log(1 - p_b)}{\log(q)} \quad \Rightarrow \quad k = \frac{\log(1 - p_b)}{\log(q)}
\end{align*}
\]

Note that \( k \) is not directly dependent to the size of the population \( P \), only on the probability of success \( p_b \). As an example, to find focused communities with at least 30% the size of population \( P \) and with success probability \( p_b = .99 \) we need at least 13 samples.

For communities with size 70% or more and the same probability \( p_b \) we need only 4 uniformly selected samples.

4.3 Sample&Climb Climbing Phase

The Climbing phase follows the Seeding phase by consuming the sampled datapoint and producing a maximally focused community. More specifically, a tuple \( t \) is received from the Seeding phase and the focus metric from Equation (2) is utilized to climb the lattice (see Figure 2) from \( t \) to a new tuple \( t' \leq t \), so that the support of \( t' \) in \( P \) is maximized and is at least \( \xi \), and \( t' \)’s focus metric remains below the relaxation threshold \( \epsilon \). Similar to hill-climbing techniques, in every new iteration a new neighbor of the current solution is generated until an acceptable solution is reached. A tuple \( t \) has \( N \) possible neighbors: each one can be reached by generalizing a different attribute value of \( t \).

4.3.1 Basic Climbing Approach

The pseudocode in Algorithm 2 describes this process. Starting from a tuple \( t \), a new neighbor is produced in every iteration till a maximally focused community or a HEAD tuple is reached. \( HEAD \) represents the unique tuple that has all of its attribute values fully generalized: \( HEAD = [*,*,...,*,...,*] \).

An accepted solution (focused community) is reached when both conditions in line 5 in Algorithm 2 are satisfied (focus metric and support). These two conditions alone do not guarantee maximality therefore the algorithm will not return at this point but will continue until the \( HEAD \) is reached and at this point will return the most recent accepted value for \( t' \). In line 7 the next attribute for generalization is selected: \( a_g \). Different selection policies will yield different results and offer different guarantees. Using the selected attribute, a new tuple \( t_{temp} \) is generated, identical to the previous \( t_{temp} \) on all attributes except \( a_g \), which gets generalized.

Since the climbing process always follows an upward path – a neighbor is created only by generalizing a single attribute – there is a well defined maximum number of iterations, equal to:

\[
\sum_{i=1}^{N} (H[i].numLevels - 1)
\]

where \( H[i].numLevels \) is the number of hierarchical levels for the \( i^{th} \) attribute. This sum can be approximated by \( O(N) \). However, the selection policy for the next attribute to generalize has a significant impact on the performance of extracting a tuple \( t' \) that eventually corresponds to a maximally focused community. We will first discuss the exact selection policy that guarantees the discovery of a maximally focused community and then propose a greedy policy for a more efficient selection.

Algorithm 2: Climbing phase

\[
\begin{align*}
\textbf{Data:} & \ Tuples \ P, \ attribute \ hierarchies H[N] \\
\textbf{Result:} & \ Maximally \ generalized \ tuple t' \\
1 & \text{begin} \\
2 & \text{t}_{temp} \leftarrow t; \\
3 & \text{t'} \leftarrow \text{NULL}; \\
4 & \text{while} \ t_{temp} \neq \text{HEAD} \text{ do} \\
5 & \quad \text{if} \ \text{focus}(t_{temp},P) \leq \epsilon \ \text{and} \ \text{support}(t_{temp},P) \geq \xi|P| \\text{then} \\
6 & \quad \quad t' \leftarrow t_{temp}; \\
7 & \quad a_g \leftarrow \text{getNextAttributeToGeneralize}(t_{temp},P,H); \\
8 & \quad \text{// Replace } a_g \text{ with parent of } a_g \text{ in } t_{temp} \\
9 & \quad t_{temp} \leftarrow \{a \in t_{temp} | a_g \Rightarrow H.parentValue(a_g)\}; \\
10 & \text{end}
\end{align*}
\]
We start with a policy for selecting the next attribute of a tuple \( t \) to generalize \((a_g)\) which guarantees reaching the correct attribute values of a maximally focused community \( C \), if one exists and \( t \geq C_t \). This policy involves choosing the attribute with a value that when generalized to the next hierarchical level results in the largest gain on the new tuple’s support.

\[
\arg\max_{a_g \in t} \text{support}(a \in t | a_g \text{.value} \leftarrow \text{H.parent}(a_g \text{.value}), P)
\]

where \( a_g \text{.value} \) is the current value of the attribute \( a_g \) (e.g. if the attribute is Location, it could be Los Angeles or California). The \( \text{argmax} \) function returns the attribute value for which the tuple support attains its maximum value. The main drawback of this approach is the need to calculate the support of \( N \) different tuples in each iteration. Since a total of \( O(N) \) iterations is required to reach a maximally focused community, the total time complexity becomes quadratic \((O(N^2))\).

**Theorem 1.** The attribute generalization policy will lead to a maximally focused community \( C \) if the starting tuple \( t \geq C_t \).

**Proof.** Let \( C \) be a maximally focused community with size \( |C| \geq \epsilon P \) and with a focus metric less than \( \epsilon \). Let \( t \) be a starting tuple with \( n \) attribute values so that \( t \geq C_t \) \((C_t \text{ can be reached by generalizing attribute values in } t)\). \( C_t \) can be correctly reached from \( t \) if after \( O(n) \) iterations \( t' \) becomes \( C_t \). The only way that a selection policy can fail to reach \( C_t \), during the climb from \( t \) to \( \text{HEAD} \), is if one attribute value of \( t \) gets generalized beyond the corresponding attribute value of \( C_t \). To prove the theorem we need to show that the selection policy will never select to generalize an attribute of \( t \) that has reached the same value with the corresponding attribute of \( C_t \).

Let \( t_i \) and \( t_j \) be the \( i^{\text{th}} \) and \( j^{\text{th}} \) attribute values of \( t \), and \( c_i \) and \( c_j \) the \( i^{\text{th}} \) and \( j^{\text{th}} \) attribute values of \( C_t \). Assume that \( t_i \) has reached the same value with \( c_i \), and that \( t_j \) has not: \( t_j > c_j \). \( C \) is a maximally focused community so given the maximality property any further generalization of an attribute in \( C_t \) cannot lead to a new focused community. Therefore, the generalization of \( t_i \) will not increase the support of \( t \) while the selection of attribute \( t_j \) (or any other attribute not generalized to the same level with \( C_t \)) will result in a new tuple \( t' \) with an increased support. Thus, as long as there are attribute values in \( t \) that are not generalized to the same level of \( C_t \), their selection will always be prioritized over attribute values that have reached the correct level of generalization, till all of them are correctly generalized.

**4.3.2 Greedy Attribute Selection Approach**

To improve the efficiency of the focused community extraction algorithm and render it scalable, we propose a greedy policy to select the attribute \( a_{g^*} \), choose the attribute value of the tuple that has the smallest support in \( P \) (\( \text{argmin} \)). The intuition behind this approach is that in a homophilic community defined by \( N \) characteristics, the characteristic with the smallest support is the one that most likely constrains the size of the community the most. More specifically, the support of a tuple \( t \) is equal to the size of the intersection of the \( N \) attribute values in \( t \) and the size of this intersection is bounded by the support of the attribute value with the smallest support. The only way to increase this bound is by generalizing the smallest attribute in order to match more datapoints. This observation is illustrated in Figure 1b: if either of the two values \( v_{a_2} \) or \( v_{a_3} \) is generalized, the intersection of the three attributes will still be limited by value \( v_{a_2} \) and remain almost the same size. Instead, the generalization of \( v_{a_2} \) has the greatest potential to increase the

**4.3.3 Climbing Phase Extension**

The Climbing algorithm, as described so far, requires that a community characteristic is precisely described through the hierarchy. However, there can be cases where a community expands in a level that is not present as a node in the hierarchy. For example, a focused community might be exclusively split in two remote locations: San Diego and New York. The only way to describe both cities with one attribute value would be to generalize them to the country level: USA. Ideally, we want to avoid this generalization since USA also indicates a community that expands in the whole country and not only San Diego and New York.

If this scenario occurs for a single attribute only, then two individual focused communities can be extracted. When this pattern is observed in more than one attributes then the extraction of individual communities might not be possible. Figure 5b illustrates such an example where a focused community resides in two separate cities and two disjoint age groups. Note that the intersection between a single location value and single age value does not form

![Figure 5: (a) An example of the specific case for which the greedy attribute selection policy can fail to select the correct attribute. (b) An example of a community with split attribute values.](image-url)
a focused community by itself (based on the definition). A focused community can only be reached if the location attribute has value SanDiego|NewYork and the age attribute has value 13–18|30+, where the “|” operator stands for “or”. We will be referring to such values as “split attribute values” and to the individual values of a split attribute value as siblings.

The generalization of an attribute value from \( v \) to \( v' \), given a population of datapoints \( P \), should be acceptable if the majority of attribute values directly succeeding \( v' \) are also represented in \( P \). For example, we can generalize the value Male to “us” only if there is a significant number of females in the population as well (at least \( \varepsilon \% \)). Or generalize the value Los Angeles to California only if the majority of cities in California also have a presence in \( P \). Calculating the presence of a majority can be computationally expensive so instead we utilize a simpler heuristic: If at least \( m \) different attribute values directly succeeding \( v' \) are also represented in \( P \) then the generalization is acceptable. In the experiments, we used a value of \( m = 3 \) for all attributes except Location which had a value of \( m = 7 \). This results in an accuracy of 93%, with the exact approach to test majority being the ground truth.

To extract communities with split attribute values the climbing algorithm has to be altered to handle such cases. The necessary changes are shown in Algorithm 3 which replace lines 7 and 8 in Algorithm 2. The intuition is that when an attribute value generalization is not acceptable there might still be other sibling values which are part of a homophilic community. By sampling a few datapoints from \( P \) we can discover those sibling values (lines 5 and 6). It is still necessary to generalize these sibling values (line 8) with regards to \( t_{temp} \) to eventually reach a maximally community. For example, let a focused community expand in two regions California and Florida and only there. Starting from the attribute value \( a_g = \) Los Angeles, we can generalize to California \( (a_g \leftarrow \) California) but generalizing California to USA is not acceptable since the whole country of 50 states can not be represented by just two states. If a new sample from \( P \) is seeded, other cities from California and Florida will be discovered as sibling values. Cities in California are already part of the current \( a_g \) and can be ignored. Cities in Florida, like Miami can be accepted but should be tested for further generalization, i.e. to Florida. Since this generalization is acceptable, Florida will become a sibling value to California.

This algorithmic extension has no significant impact on the complexity. Though the cases of split attribute values were quite rare in our experiments, even when met, the additional complexity only includes the additional steps for checking and generalizing the new attribute siblings (line 8 in Algorithm 3). Given that there can be a maximum of \( 1/\varepsilon \) siblings for each attribute the total complexity becomes \( O(n/\varepsilon) \), with \( \varepsilon \leq 1 \).

Algorithm 3: Extension for handling split attribute values

Result: Maximally generalized tuple \( t' \) with split attributes

1. If \( t' \neq NULL \) and \( focus(t_{temp}, P) > \varepsilon \) then
2. \( a_g \leftarrow get\text{Next}\text{AttributeToGeneralize}(t_{temp}, P, H) \);
3. If generalization of \( t_{temp}[a_g] \) not acceptable then
4. \( S' \leftarrow sample(P, H) \);
5. \( splitValues \leftarrow \{t'[a_g] | t \in S' \text{ and } t \geq t_{temp}\} \);
6. For \( splitValue \in splitValues \) do
7. \( splitValue \leftarrow generalize(splitValue) \);
8. Else
9. \( splitValues \leftarrow \{H\text{.parentValue}(a_g)\} \);
10. \( t_{temp} \leftarrow \{a \in t_{temp} | a_g \leftarrow splitValues\} \);
11. End

4.4 Topic Ranking Phase

During the traversal of datapoints using a sliding window of size \(|W|\), the Sample&Climb module identifies a set of topics with their corresponding maximally focused communities. This set can be quite large and may include uninteresting cases. We therefore developed a score metric to order this set in a way that ensures more interesting or surprising topics are ranked higher. Initially, a filtering pre-step is necessary to remove topics with unsurprising focused communities. For example, if a topic has a focused community of males with support 50%, and half of the total Twitter population in \( W \) is male anyway, then the focused community is actually expected. We perform a Pearson’s chi-squared test to compute the anomaly of a topic’s community (expected Vs. observed) and further filter-in candidate topics with significance of \( p \leq 0.05 \).

To obtain an interesting ranking of topics, after filtering out the topics with expected focused communities, we use a combination of two measures: Inverse Community Frequency and Relative Community Popularity. Both measures aim to normalize the raw frequency of a topic in order to boost those topics with interesting focused communities. Inverse Community Frequency (icf) is inspired by Inverse Document Frequency from text document ranking in Information Retrieval. Here we use it in a similar context: to tune down community characteristics that get associated with many topics. A community characteristic that appears in few topics only should be more interesting. Inverse Community Frequency, measures how many topics in the whole window \( W \) of datapoints also share a community characteristic. For example, the icf of location Santa Barbara will depend on how many topics in \( W \) have a focused community that contains Santa Barbara. The icf score of a community \( C \) is the product of icfs scores for each attribute value in \( C \). The icf score for a single attribute value \( a \) is equal to:

\[
\text{icf}(a) = \log \frac{N_t}{|\{ t \in W | a \in C\} |}
\]

where \( N_t \) is the total number of topics in \( W \) and the fraction denominator is equal to the number of topics \( T \) in \( W \) with a community \( C \) that contains the attribute value \( a \). Relative Popularity takes values between 0 and 1 and practically compares the size of a topic’s focused community with the size of the community with the same characteristics in the window \( W \) of datapoints. The relative popularity score is calculated as the fraction of the support of a community in \( P \) over the support of the community in \( W \):

\[
\text{rp}(C) = \frac{\text{support}(C, P)}{\text{support}(C, W)}
\]

For example, if a topic is being discussed by 100 women and the number of women in \( W \) is also 100, then this community has a relative popularity of 1. The overall scoring function is based on each topic’s extracted focused community \( C \) and uses both notions of relative popularity and exclusive focus:

\[
\text{score}(T) = P \times \text{rp}(C) \times \text{icf}(C)
\]

where \( C \) is a focused community of the topic \( T \), \( P \) is the population of the topic. The overall score of a topic is proportional to the topic’s raw popularity (size of \( P \)), the relative popularity score of the topic’s community, and the icf score of its community. Using this score metric we rank the candidate topics and obtain a list of top-k Trending Topics as the final result of the algorithm.

5. HANDLING MISSING VALUES

One of the main challenges when dealing with real social data sets is the sparsity of attribute values. This observed sparsity (missing values) is due to the low recall of specific inference tasks which usually originates in the general lack of sufficient information to
infer attributes with high confidence (e.g., not enough textual information to infer the age of a user). For example, it is possible that a topic is mentioned by 1000 users, and within this population 100 have complete location information and 250 have their age successfully inferred. The intersection of the users that have both location and age values extracted is even smaller.

Let $P$ be a population of datapoints and $C$ a maximally focused community in $P$ if there were no missing values. Let $N$ be the number of attributes and $m_i \in [0,1]$ the portion of missing values for each attribute $a_i$. The discovery of a focused community requires at least one attribute value (exclusive feature) with size equal to the size of the community, assuming 0 relaxation ($\epsilon = 0$). Note that the size of the community is equal to the number of datapoints in $P$ that match the tuple that defines the community. In the presence of missing values, an attribute tuple will not match every datapoint that it should. For example, the tuple [California, Male, *] does not match the datapoint [Los Angeles, ∞, 18] because $\bot$ does not succeed Male. Therefore, if there are randomly missing values in each attribute, the observed size of the community $C$ and the size of the exclusive feature will differ and the focus metric will not be low enough to identify the focused community.

To overcome this problem, we allow a tuple to match missing values during counting. Referring back to the previous example, we allow the tuple [California, Male, *] to match the datapoint [Los Angeles, ∞, 18]. This alteration fixes the issue of under-counting a tuple, but introduces over-counting: additional datapoints are now counted as part of a community. However, the community size over-estimation is statistically bounded. Let $v_f$ be the attribute value that plays the role of the exclusive feature in the focused community $C$ and let $m_f$ be the ratio of missing values for the attribute $a_f$. The focused community can be divided in two parts: the datapoints that belong in the community and have a value $v_f$ for the attribute $a_f$ and the datapoints that belong in the community and have a value $\bot$ for the attribute $a_f$ (missing value). Similarly, the datapoints outside the focused community can be divided in two parts: the datapoints that have a value $v_f' \neq v_f$ for the attribute $a_f$ and the datapoints that have a value $\bot$ for the attribute $a_f$ (missing value). Note that there are no datapoints outside the community with value $v_f$ for the attribute $a_f$ based on the definition of the focused community. The datapoints that could be mistakenly counted are the ones outside the community, with a missing value. Among these datapoints, the other $N - 1$ attribute values must also match the corresponding attribute values of the community or be missing. The expected size of this subset of datapoints is bounded by:

$$m_f |P - C| \leq m_f |P| - \xi |P| = m_f (1 - \xi) |P|$$

This is an upper bound of the expected count of wrongly counted datapoints since some of these datapoints will likely have a different non-missing value for some attribute $a_i \neq a_f$ and will not match to the community’s tuple. In any case, in the presence of many missing values it is recommended to use a higher support threshold $\xi$ for the correct detection of focused communities since the above value gets closer to 0 when $\xi \to 1$.

### 6. EXPERIMENTS AND ANALYSIS

To understand the performance and accuracy of the proposed algorithm we performed experiments on two different datasets, one real and one synthetic. The real dataset comprises of a 10% sample of the Twitter stream from 2013 and 2014 and tested the performance of the algorithm and quality of the results. The synthetic dataset was specifically constructed to examine the accuracy and recall of the proposed approach and includes a complete spectrum of scenarios — some that might be rare in a real dataset.

We introduce the details of the Twitter dataset and attribute extraction process in Subsection 6.1, then offer a qualitative analysis of the produced results in Subsection 6.2. In Subsection 6.3 we offer a quantitative evaluation of the Twitter results to measure their usefulness and attractiveness. Finally, we discuss the performance of the algorithm on synthetic data in Subsection 6.4.

#### 6.1 The Twitter Dataset

The used Twitter dataset contains a uniform 10% sample of all the tweets and Twitter users from the following two periods: September 12 to October 26 of 2013 (45 days) and April 16 to May 24 of 2014 (39 days). The pool of topics contains every mentioned hashtag or capitalized entity from the tweets’ raw text. The extracted tweet features include location, the list of external user mentions (@-replies), the device the tweet was posted from (e.g. iPhone, Android, web browser), and the general sentiment. Location extraction was done on (1) the tweet level using Twitter’s geo-tagging mechanism, and to further improve the recall, on (2) the user level using a user-provided raw text field (similarly to [27, 2]). To infer location based on the user’s field we applied a simple but precise pattern matching process that could identify location patterns like: “City, Region, Country”, or “Region, Country”, or just “Country”. To validate the patterns we used a Location hierarchy provided by the MaxMind database [12]. The user device was extracted from the available information provided by the Twitter API [26]. To infer the sentiment of a tweet we used the SentiStrength tool [24]. Note that not all features were available in every tweet; for example, less than 2% of the tweets had an explicit location tag or non-neutral sentiment (missing values).

Meaningful and interesting community extraction requires a diverse set of user characteristics/demographics. To expand the number of extracted attributes from the Twitter dataset we additionally infer the users’ age, gender, political affiliation, and sports team preference. To extract gender and age we applied existing language models extracted from Schwartz et al. [23] on social media data. To apply the models we gathered all the tweets of every user for each of the two analyzed periods of data. While this is an expensive process, especially space-wise, it can be done offline and does not affect the complexity of our Sample&Climb algorithm. For political affiliation we gathered the Twitter accounts associated with the three most popular US political parties: Democrats, Republicans, and Libertarians. Then, a user’s political affiliation was determined based on their interactions (@-replies) with these accounts. Similarly for sports, we collected the Twitter accounts of teams, players, and coaches for the following four US professional sports: Baseball, Basketball, Football, and Hockey. For every sport, a user’s team preference was inferred based on their interactions with each team’s accounts. Table 1 shows the precision of each inference task. Given that the language models are in English, age and gender inference only works for English speaking users. Similarly, political affiliation and sports teams are focused on users within the United States and Canada. Precision was calculated manually on a sample of 100 users from each category. However, for the age and gender inference we list the calculated precision from Schwartz et al. Note that their models were tested on Facebook data and not on Twitter, so precision might differ slightly.

In total, the experimental setup contained 10 attributes: 1) Location (either from the tweet or the user), 2) Age, 3) Gender, 4) Political affiliation, 5) Baseball team, 6) Basketball team, 7) Football team, 8) Hockey team, 9) Tweeting device (e.g. iPhone), and 10) Sentiment. While the sentiment is not strictly a user characteristic, it helps with the interpretation of the results by hinting at the attitude of the community towards the topic. Apart from Loca-
tion, and Device all attribute hierarchies have only 2 levels (trivial hierarchy). Specifically, the Location hierarchy has 4 levels: city, region, country, and "\*". The Device hierarchy has 3 levels: specific device, "mobile"/"desktop", and "\*".

**Settings.** The execution of the community extraction algorithm was applied on the stream of tweets using a sliding window of size 500,000. On a typical day this amount of tweets can be produced within two minutes of real time. For every new window new topics get introduced, existing topics receive additional mentions, and old topics get evicted. To reduce noise, candidate topics are required to have at least 50 mentions during the window. The rest of the algorithm settings are: selection policy: greedy, seeding sample size \(\xi\): 20 datapoints, \(\epsilon\): 0.15, \(\xi\): 0.3. The choice of \(\epsilon\) is based on the fact that Twitter data is noisy and the community extraction should be relaxed enough to accommodate this noise. The value of the support threshold \(\xi\) is based on the average population of a Trending Topic on Twitter, which is usually between 1K and 200K tweets, therefore we can expect communities of size between 300 and 60K users (smaller communities would not be interesting). Note that our dataset only contains 10% of the tweets in Twitter Firehose, therefore all reported size numbers in the results section should be 1/10 of what we would get on the whole Twitter dataset.

### 6.2 Qualitative Evaluation of Twitter Results

The final product of this experiment, is an ordered list of Trending Topics and the corresponding maximally focused communities that were extracted, in each window. The extracted community of a topic might differ between different windows as different or additional users mention the topic and the population changes.

We cherry-picked some topics to showcase interesting behaviors and qualitatively argue that the results actually make sense. These topics are listed in Tables 2 (general interest trends) and 3 (trends with a sports related focus). A "\*\*" value indicates that the attribute got generalized to its top level of the hierarchy. A "*" value indicates that there was not enough information to extract a specific attribute value (due to missing values). Attribute values for Device and Basketball team are omitted due to lack of space. Topics that appear twice are taken from different windows, or even days, and are listed to show the dynamic nature of focused communities as the topic population grows or just changes.

One interesting result includes hashtag #DisneySide which was a social media campaign by US Disney Parks. Disney asked fans to tweet photos of their 'Disney Side' from their visit to a Disney theme park. During the first day, most of the tweets occurred in the two cities where a Disney park is located: Anaheim, California and Orlando, Florida (split values). The next day, the campaign audience expanded to the include whole states of California and Florida.

### 6.3 Quantitative Evaluation of Twitter Results

As with many unsupervised learning tasks, evaluating the produced results is a challenging task. In content recommendation systems used by real users, one can run A/B tests to compare the success of the algorithm with a baseline. To evaluate the community-based topics in terms of potential usefulness and interestingness we (a) measure the entropy of the results as an objective quantitative measure, and (b) asked human evaluators to choose their favorite topics from a mixed pool of trending topics.

Using the notions of Self-information and Entropy from Information Theory we provide a measure of the information content for community-based trending topics. Self-information can capture how surprising an event is based on the probability of the event.
Then, we can calculate the entropy of the experiment (extracting community-based trending topics) as the expected value of every trending topic’s self-information. The self-information of the community \( C_T \) for a single topic \( T \) is \( I(C_T) = -\log_2(Prob(C_T)) \). Intuitively, the less likely a community is to observed the higher its self-information. The self-information of the community-based trending topics can be measured in bits. The average entropy for frequency-based topics was found to be 0.27 bits. This indicates that the extracted topics using our method contain surprising and potentially useful communities that cannot be trivially anticipated or that are not observed in topics ranked by raw frequency.

To measure exactly how much more interesting real users would find the extracted topics we performed the following experiment with human evaluators (colleagues and Amazon Turkers): We randomly picked a timeslot from our dataset and created a pool of 10 topics, which included a random selection of topics ranked by frequency-based topics by our algorithm and topics denoted by (F) are frequency-based. All of the top selections ended up being community-based topics, which included a random selection of topics ranked by raw frequency.

We ran 10 topics on 12 Computer Science graduate students from the University of California, Santa Barbara. Topics denoted by (C) are community-based and topics denoted by (F) are frequency-based. We asked each evaluator to pick 5 topics (in no particular order) that they find the most interesting, where a topic is defined as interesting to a user if they would like to know more about it; get tweets that contain it, read news articles, etc.

The histogram in Figure 6 shows the results of this experiment (similarly to Table 4). These 10 topics were then presented, unlabeled and randomly ordered, to an evaluator. We asked each evaluator to pick 5 topics (in no particular order) that they find the most interesting, where a topic is defined as interesting to a user if they would like to know more about it; get tweets that contain it, read news articles, see related images, etc.

The histogram in Figure 6 shows the results of this experiment performed on 12 Computer Science graduate students from the University of California, Santa Barbara. Topics denoted by (C) are community-based and topics denoted by (F) are frequency-based. All of the top selections ended up being community-based topics even though some evaluators were interested in simply popular topics like #ThrowbackThursday or #NowPlaying.
To reduce any bias on the reported evaluations results, we performed the same experiment with 5 topic pools on Amazon Turk with an average of 72 participants per pool and 10 topics per pool. All participating turkers were based in the US. The results are shown in Table 5. The first two rows display the percentage of community-based topics (c-topics) in the top-3 and top-5 of the evaluators’ selections. On average, community-based topics were ranked in the top-3 on 73.3% of the times, and in the top-5 on 60% of the times. These two values indicate that in both the top-3 and top-5 cases, the majority of interesting topics was community-based. The final two rows of Table 5 show the percentages of c-topic and frequency-based topic (f-topic) selections — how many times an evaluator clicked a topic of each category as interesting. This value can also be viewed as the probability of each category/method to produce an interesting topic. On average, community-based topics have 26.86% better chance to be more interesting than raw frequency ranked topics, which shows that in most cases users will find our algorithm’s results more interesting. Overall, a few topics ranked by raw frequency will still be interesting to users due to their popularity, but overall our method delivers more appealing results to the user. The more rare extended version of a community where attribute values are split (like in Figure 5b), we altered 1000 cases so that the selected values in each attribute dimension were more than one; half of the times under the same direct ancestor in the hierarchy and the other half with separate direct ancestors. We limited the number of split attributes of a community to a maximum of 5. The algorithm settings that we used on the synthetic dataset were the same as with the Twitter experiments: selection policy: greedy, seeding sample size: 20 datapoints, \( \epsilon : 0.15 \), \( \xi : 0.3 \).

### Results

The algorithm was able to find the correct communities in each synthetic population with an accuracy of 93.1%. A community extraction was labeled as successful when the exact correct community could be identified. In the rest of the cases that failed, most of the time there would be a community attribute value or two that were further generalized than expected. By measuring the ratio of correctly extracted attribute values for each community the average accuracy is 97.2%. The running time for all 10000 cases was a little less than 10 minutes on a 2.6GHz Inter Core i5 workstation.

### 7. CONCLUSIONS

We study the problem of extracting multi-dimensional communities focused on individual topics by introducing the notion of a maximally focused community with properties that enable the efficient discovery of interested communities defined by a subset of social attributes. These properties led to the implementation of an algorithmic framework for the extraction of maximally focused communities of any topic with proved linear time complexity. Finally, we provide a robust ranking that boosts topics with relatively popular or exclusively focused communities through metrics adapted from Information Retrieval.

Extensive experimentation was conducted on two different datasets: one real from Twitter with data from large periods in 2013/14 and one synthetic. The results highlight the efficiency, correctness, and stability of our proposed algorithm. We are able to identify interesting communities for Trending Topics, sometimes expected and sometimes unexpected. It is interesting to observe that females in Boston, which also support the Democratic party, show their solidarity to an arrested teen (#FreeJustina). It is unexpected to discover the hijacking of a touristic hashtag in Egypt from local citizens that try to raise awareness for the country’s political situation (#ReasonsToVisitEgypt). Such data can be used by advertisers, analysts, the police, or any other entity, to better understand a topic’s participating population or who contributes to a trend.

### 8. REFERENCES

Table 5: Evaluation results from Amazon Turk on 5 different pools of topics.

<table>
<thead>
<tr>
<th></th>
<th>Pool 1</th>
<th>Pool 2</th>
<th>Pool 3</th>
<th>Pool 4</th>
<th>Pool 5</th>
<th>All Pools Average</th>
<th>Local (Pool 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>% of c-topics in top-3</td>
<td>33.3%</td>
<td>100%</td>
<td>66.6%</td>
<td>100%</td>
<td>66.6%</td>
<td>73.3%</td>
<td>100%</td>
</tr>
<tr>
<td>% of c-topics in top-5</td>
<td>60%</td>
<td>60%</td>
<td>40%</td>
<td>80%</td>
<td>60%</td>
<td>60%</td>
<td>100%</td>
</tr>
<tr>
<td>% of clicks on c-topics</td>
<td>49.75%</td>
<td>54.86%</td>
<td>52.28%</td>
<td>64%</td>
<td>58.75%</td>
<td>55.92%</td>
<td>70%</td>
</tr>
<tr>
<td>% of clicks on f-topics</td>
<td>50.25%</td>
<td>45.14%</td>
<td>47.71%</td>
<td>36%</td>
<td>41.25%</td>
<td>44.08%</td>
<td>30%</td>
</tr>
</tbody>
</table>


