Querying Term Associations and their Temporal Evolution in Social Data[*]

Vassilis Plachouras
IMIS / RC "ATHENA"
Artemidos 6
Athens 15125, Greece
vplachouras@imis.athena-innovation.gr

Yannis Stavrakas
IMIS / RC "ATHENA"
Artemidos 6
Athens 15125, Greece
yannis@imis.athena-innovation.gr

ABSTRACT

There is a growing number of applications that rely on data collected from online social networks. These applications typically issue search requests for keywords and process the data returned by online social networks through APIs. The selection of keywords can have an important impact on the quality of the results and the appropriateness of the collected data for further analysis. Indeed, adding or removing keywords in the search requests may change the characteristics of the sampled data. Hence, it is important for users to have the ability to explore data and to express complex requests in order to discover the context of collected data. In this work, we propose a model and a number of query operators that allow users to select data and explore its context by means of querying for associations between keywords or entities as well as their evolution over time. The model supports different time granularities and the calculation of term association weights based on the context of terms. We demonstrate the use of the model and the query operators with a running example based on data we have collected from the microblogging service Twitter, and a first implementation running on top of a relational database.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]

General Terms

Experimentation, Design

Keywords

Social networks, term associations, temporal evolution, query operators

[*]This work is partly funded by the European Commission under ARCOMEM (ICT 270239).

1. INTRODUCTION

With the growth of online social networks and microblogging sites, such as Facebook[1] or Twitter[2], and the availability of their data through APIs, there is a increasing number of applications that consume social network activity data. For example, Asur and Huberman [1] correlate the volume of chatter in Twitter about movies with their box office revenues. They collect data by searching for terms present in the movie titles. Jansen *et al.* [6] examine how Twitter users express their opinions about brands. For each examined brand, they query a service, which analyzes attitudes expressed in tweets. Sakaki *et al.* [12] search for tweets containing words such as *earthquake* and *shaking* to track earthquakes.

All the applications mentioned above issue search queries to the API of Twitter for specific terms that relate to the application domain. The selection of terms, however, can have an important impact on the quality of the results. O'Connor *et al.* [11] observe such an impact when they use variations of words to correlate a sentiment score from tweets with an index of consumer confidence. The same issue is also mentioned in the context of sentiment analysis by Jiang *et al.*, who suggest that classifying the sentiment expressed about a target keyword may be enhanced by also classifying the sentiment expressed for other related keywords [7]. Consequently, we argue that it is important to be able to explore the context of terms, to create datasets and to perform complex analysis tasks based on term associations. Currently, there is a gap between this need and the functionality that online social network APIs offer.

In this work, we introduce a model and a number of query operators that enable users to explore the context of keywords in data obtained from online social networks and microblogging services. The query operators allow users to discover the relationships between keywords (as they are implied by their occurrence in tweets) and their evolution in time. They also allow to express complex conditions on the associations of terms that take temporal aspects into account and to retrieve the subset of tweets that satisfy these conditions. For example, a journalist, who is exploring Twitter data consisting of timestamped hashtags, can issue the following query concerning the financial crisis:

> For the period during which there is a strong association between hashtags #crisis and #protest,

[1]`http://www.facebook.com/`
[2]`http://www.twitter.com/`

which other hashtags are associated to both #crisis and #protest? Which are the relevant tweets?

The above example query will first identify the timespan during which there is a strong association between the hashtags #crisis and #protest. Next, other hashtags which are strongly related to both #crisis and #protest will be identified. Finally, the relevant tweets that contain any of the identified hashtags in the corresponding timespan will be returned.

We assume that two terms are associated when they co-occur in the same tweet. However, the association between terms is not only indicated by their co-occurrence. If term n is associated to term c and c is associated also to term d, then we can assume that n is to some extent associated to d as well. Thus, we define the model and the query operators so as to allow for the inference of new associations between terms that do not co-occur in tweets.

The remainder of this work is organized as follows. Section 2 describes the introduced model. Section 3 presents the query operators and gives an illustrative example for each one. We present a running example on data we have collected from Twitter in Section 4. Finally, we close with a review of related work in Section 5 and some concluding remarks in Section 6.

2. TEMPORAL TERM ASSOCIATION MODEL

In this section, we provide a formal definition of the model. We first model the temporal associations between terms as a set of quintuples. We also show that the set of quintuples can be viewed as a labeled multidigraph. Next, we define how associations between terms are calculated.

The model can be applied to any temporally evolving collection of documents. In the context of this work, we assume that documents are tweets, which are downloaded at regular intervals. The downloaded tweets are processed at regular time instances $t = 1, 2, \dots, i$. At the i-th time instance, we populate with the tweets downloaded between instances $i-1$ and i the relation TT (which stands for timestamped terms), with attributes *id*, *pt* and *term*. The attribute *id* is a unique identifier for each tweet. The attribute *pt* denotes the publication time of the tweet. The attribute *term* is any word or short phrase of interest that occurs in the corresponding tweet, including hashtags or the output of processes such us entity recognition. The relation TT may actually contain more attributes than the basic three mentioned here. We also store the actual content and metadata of tweets, in the form received from the Twitter API, and indexed by the unique identifier *id*.

Once we collect, process tweets and update the relation TT for a time instance t, we add a new snapshot of the model for t. The model comprises all successive snapshots. Note that the publication times of tweets are distinct from the time instances, which are represented as successive integers.

2.1 Model definition

We define the model $\mathcal{M}$ as a set of quintuples:

$$\mathcal{M} = \{\langle n, c, w, T, g\rangle | n, c \in V, w \in [0,1], T \in 2^{\mathbb{Z}+}, g \in \mathbb{Z}+\}$$

where n is a *target node* and c is a *context node*. Both n and c correspond to terms existing in the relation TT with publication times matching time instances in T. V denotes the set of distinct terms existing in relation TT. w is a real number, which quantifies the strength of association from n to c. T is a set of integers. A shorthand is used for expressing sets of consecutive time instances; for example $\{1 \dots 3, 9 \dots 12\}$ is equivalent to the expanded notation $\{1, 2, 3, 9, 10, 11, 12\}$. The time granularity g dictates how to aggregate and interpret the elementary timespans as days, weeks, etc. For example, if the elementary time instance in T corresponds to 4 hours and g=6, then the time granularity is 24 hours (one day).

The intuition behind the model is that n is associated to c with weight w for the set of time instances in T interpreted according to the time granularity g.

Hereafter, we define the association weight as the probability to observe c given that we have observed n in the tweets published in the time instances $t \in T$:

$$w = P_T(n \to c) = \frac{\sum_{n,c \in tw} \frac{1}{|tw|-1}}{\sum_{n \in tw} 1} \qquad (1)$$

The sum in the nominator is computed for all tweets in timespan T where n and c co-occur. The sum in the denominator is computed over all tweets in timespan T that contain n. We denote by $|tw|$ the number of terms in a tweet. The intuition of Equation 1 is that the importance of the co-occurrence between terms is equally split among all co-occurring terms. Moreover, Equation 1 captures the case where a term may be frequent, but most of the times occurs without any other terms. In this case, the denominator $\sum_{n \in tw} 1$ will be higher and consequently the weight $w = P_T(n \to c)$ will be lower. When a term occurs on its own for some tweets in T, then we add a quintuple where $n = c$ with w equal to the number of times that the n appears on its own over the total frequency of n in timespan T. We note that the definition for the weight w is only one of the possible approaches we could employ. Other definitions may include the log-likelihood ratio [5], or the correlation between the frequency time series of term occurrences. The definition we have used in this work is appropriate for our setting because it captures the probability to observe c given that we have observed n, and corresponds to the transition probabilities of a first-order Markov Chain.

As mentioned earlier in this section, the model we have defined as a set of quintuples can also be viewed as a labeled multidigraph, where the set of vertexes corresponds to the set of terms V existing in relation TT and the set of directed edges correspond to the ordered pairs (n, c) of target and context nodes from the quintuples of the model. Each edge is labeled with the weight w, the timespan T and the time granularity g. As an example, consider for time instance $t = 1$ the tweets $tw_1 = \{a\}$, $tw_2 = \{a\}$, $tw_3 = \{a, b\}$, $tw_4 = \{a, c\}$, $tw_5 = \{c\}$ and for time instance $t = 2$ the tweets $tw_6 = \{a, c\}$ and $tw_7 = \{a\}$, where sets contain the tweet terms. Then, the corresponding model $\mathcal{M}$ is:

$$\begin{aligned}\mathcal{M} = \{&\langle a, b, 0.25, \{1\}, 1\rangle, \langle a, c, 0.25, \{1\}, 1\rangle,\\ &\langle b, a, 1.00, \{1\}, 1\rangle, \langle c, a, 0.50, \{1\}, 1\rangle,\\ &\langle a, a, 0.50, \{1\}, 1\rangle, \langle c, c, 0.50, \{1\}, 1\rangle,\\ &\langle a, c, 0.50, \{2\}, 1\rangle, \langle c, a, 1.00, \{2\}, 1\rangle,\\ &\langle a, a, 0.50, \{2\}, 1\rangle\}\end{aligned}$$

Figure 1 shows the graph representation of model $\mathcal{M}$. There are nine edges. For each edge, the label corresponds to the weight, the set of time instances for which the edge is valid and the granularity of the corresponding set of time instances. Moreover, the graph has edges with the same source and destination nodes, denoting that a term occurred on its own in a tweet.

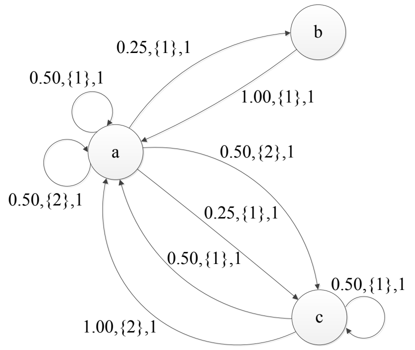

Figure 1: Representation of the model $\mathcal{M}$ as a labeled multidigraph.

3. QUERY OPERATORS

In order to be able to query the temporal term association model defined in the previous section, we introduce a set of operators and provide illustrative examples.

3.1 Filter operator

The *filter* operator receives a model $\mathcal{M}$ and a condition *cond* and it returns the set of quintuples of $\mathcal{M}$ that satisfy the given condition. The condition is specified with respect to any of the elements of the quintuple. The filter operator is written $\mathcal{M} = filter(\mathcal{M}, cond)$.

An example using the filter operator is the query $\mathcal{M}_2 = filter(\mathcal{M}_1, T \text{ inside } \{5 \ldots 12\} \wedge w \in top(10))$ which returns a model $\mathcal{M}_2$ comprising the quintuples of $\mathcal{M}_1$ for which the timespan is contained in timespan $\{5 \ldots 12\}$ and their weight is in the set of the 10 highest weights.

3.2 Fold operator

The *fold* operator receives as input a model $\mathcal{M}$ and a positive integer g, and it is written $\mathcal{M} = fold(\mathcal{M}, g)$. If g_i and g_o correspond to the granularity of the input and the output models, respectively, then $g = g_o/g_i$.

Next, we give an example of the fold operator. We assume that the input model is the following:

$$\begin{aligned}\mathcal{M}_1 = \{&\langle n_1, c_1, w_1, \{1\}, 1\rangle,\\ &\langle n_1, c_1, w_2, \{2\}, 1\rangle,\\ &\langle n_1, c_1, w_3, \{3\}, 1\rangle,\\ &\langle n_2, c_1, w_4, \{1\}, 1\rangle,\\ &\langle n_2, c_1, w_5, \{4\}, 1\rangle\}\end{aligned}$$

The operation $\mathcal{M}_2 = fold(\mathcal{M}_1, 3)$ returns the model $\mathcal{M}_2$:

$$\begin{aligned}\mathcal{M}_2 = \{&\langle n_1, c_1, w_6, \{1, 2, 3\}, 3\rangle,\\ &\langle n_2, c_1, w_4, \{1, 2, 3\}, 3\rangle,\\ &\langle n_2, c_1, w_5, \{4, 5, 6\}, 3\rangle\}\end{aligned}$$

Assuming that weights are computed as defined in Equation 1, the weight w_6 is computed as follows:

$$w_6 = P_{\{1,2,3\}}(n_1 \rightarrow c_1)$$

The weights for the quintuples where $n = n_2$ do not change, because there is no change in the number of tweets in which $n = n_2$ and $c = c_1$ for timespans $\{1, 2, 3\}$ and $\{4, 5, 6\}$.

3.3 Jump operator

The *jump* operator performs a *self join* and derives new associations between nodes. The *jump* operator receives a model as input and returns a model with expanded contexts and updated association weights. For each timespan in the input model, we compute the updated weights by creating a first-order Markov chain from the corresponding directed graph. The probability of following a path of length k between any two nodes in the Markov chain corresponds to the updated association weights between the two nodes. We assume that $k > 1$, because if $k = 1$ then the transition probabilities are equal to the association weights stored in the quintuples. The *jump* operator does not alter the set of time instances T or the granularity g of the model. The *jump* operator is written as $jump(\mathcal{M}, k)$.

Next, we give an example of the *jump* operator. We assume that $\mathcal{M}$ is the model corresponding to the directed multigraph in Figure 1. The query $\mathcal{M}' = jump(\mathcal{M}, 2)$ returns $\mathcal{M}'$ where the association weights are computed as follows. For each of the two time instances in $\mathcal{M}$, we create a Markov Chain. For time instance $t = 1$, the transition matrix is the following:

$$P_{\{1\}} = \begin{pmatrix} 0.50 & 0.25 & 0.25 \\ 1.00 & 0.00 & 0.00 \\ 0.50 & 0.00 & 0.50 \end{pmatrix}$$

where $p_{\{1\}}(1, 2)$ corresponds to the weight $P_{\{1\}}(a \rightarrow b)$ and $p_{\{1\}}(1, 3)$ corresponds to the weight $P_{\{1\}}(a \rightarrow c)$. We note that $p_{\{1\}}(1, 1)$ corresponds to the probability that a does not co-occur with other terms and it is interpreted as remaining in the same state. To compute the updated association weights for $k = 2$, we compute $P^2_{\{1\}}$. Then, the association weight from term a to term b at time instance $t = 1$ is equal to $p_{\{1\}}(1, 2) = 0.125$. After computing the association weights for time instance $t = 2$ in a similar way, the output model $\mathcal{M}'$ is the following:

$$\begin{aligned}\mathcal{M}' = \{&\langle a, a, 0.625, \{1\}, 1\rangle, \langle a, b, 0.125, \{1\}, 1\rangle,\\ &\langle a, c, 0.250, \{1\}, 1\rangle, \langle b, b, 0.250, \{1\}, 1\rangle,\\ &\langle b, a, 0.500, \{1\}, 1\rangle, \langle b, c, 0.250, \{1\}, 1\rangle,\\ &\langle c, a, 0.500, \{1\}, 1\rangle, \langle c, b, 0.125, \{1\}, 1\rangle,\\ &\langle c, c, 0.375, \{1\}, 1\rangle, \langle a, a, 0.750, \{2\}, 1\rangle,\\ &\langle a, c, 0.250, \{2\}, 1\rangle, \langle c, a, 0.500, \{2\}, 1\rangle,\\ &\langle c, c, 0.500, \{2\}, 1\rangle\}\end{aligned}$$

It is also possible to compute the transition probabilities without specifying a maximum number of transitions. In order to guarantee the convergence of the computation, we can employ a damping factor, similar to the PageRank algorithm [3].

3.4 Merge operator

The *merge* operator is similar to the *fold* operator in that it aggregates quintuples according to the context and target nodes, but without changing the time granularity g. The *merge* operator receives an input model and returns a model with the merged quintuples. If the input model $\mathcal{M}_1$ is the following:

$$\begin{aligned}\mathcal{M}_1 = \{&\langle n_1, c_1, w_1, T_1, g\rangle, \langle n_2, c_1, w_2, T_1, g\rangle,\\ &\langle n_1, c_1, w_3, T_2, g\rangle, \langle n_2, c_1, w_4, T_2, g\rangle,\\ &\langle n_3, c_1, w_5, T_2, g\rangle, \langle n_1, c_1, w_6, T_3, g\rangle,\\ &\langle n_4, c_1, w_7, T_3, g\rangle\}\end{aligned}$$

then the output model $\mathcal{M}_2 = merge(\mathcal{M}_1)$ is the following:

$$\begin{aligned}\mathcal{M}_2 = \{&\langle n_1, c_1, w_8, T_1 \cup T_2 \cup T_3, g\rangle,\\ &\langle n_2, c_1, w_9, T_1 \cup T_2, g\rangle,\\ &\langle n_3, c_1, w_{10}, T_2, g\rangle,\\ &\langle n_4, c_1, w_{11}, T_3, g\rangle\}\end{aligned}$$

The weights w_8, w_9, w_{10}, w_{11} in the quintuples of the output model $\mathcal{M}_2$ are calculated in the same way as described in Section 3.2 for the *fold* operator.

3.5 Join operator

The *join* operator performs a self-join on the input model in order to select only a part of the model that satisfies a condition matching another part of the model. Similarly to the semi-join of relational algebra, the *join* operator receives two models, $\mathcal{M}_1$ and $\mathcal{M}_2$ as input and a condition *cond* on variables of the two input models, and it returns a model $\mathcal{M}_3$ which is a subset of $\mathcal{M}_1$. The *join* operator is written as $\mathcal{M}_3 = join(\mathcal{M}_1, \mathcal{M}_2, cond)$.

Next, we give one example of applying the *join* operator. Given the model $\mathcal{M}_1$:

$$\begin{aligned}\mathcal{M}_1 = \{&\langle n_1, c_1, 0.5, \{1,2\}, 1\rangle, \langle n_1, c_2, 0.5, \{1,2\}, 1\rangle,\\ &\langle n_1, c_1, 0.7, \{3,4\}, 1\rangle, \langle n_1, c_2, 0.3, \{3,4\}, 1\rangle\}\end{aligned}$$

the query, which asks for the quintuples where their weight has increased compared to the past:

$$\begin{aligned}join(&\mathcal{M}_1 \text{ as } m, \mathcal{M}_1 \text{ as } m',\\ &m.n = m'.n \wedge m.c = m'.c\\ &\wedge min(m.T) > max(m'.T)\\ &\wedge m.w > m'.w)\end{aligned}$$

returns the model $\mathcal{M}_2 = \{\langle n_1, c_1, 0.7, \{3,4\}, 1\rangle\}$.

4. RUNNING EXAMPLE

In this section, we describe an extended running example of using the model and query operators we have defined in Sections 2 and 3, respectively, on a large sample of data we have collected from Twitter. The objective of the example is to identify the hashtags for which the strength of association with a given hashtag is increasing over time. Section 4.1 describes the dataset we have collected and Section 4.2 provides a description of the steps performed to answer the example's query using a relational database.

4.1 Dataset

Using the Streaming API of Twitter, and since we were interested only in Greek tweets, we collected a data set of tweets by tracking a set of 74 Greek stop-words. We collect tweets over a period of three months, starting from March 20, 2012 until June 20, 2012. In total, we have collected 16.5 million tweets, from which 1.2 million tweets contain at least one hashtag. Figures 2 and 3 show the daily volume of tweets and tweets containing at least one hashtag, respectively, collected over the period of three months.

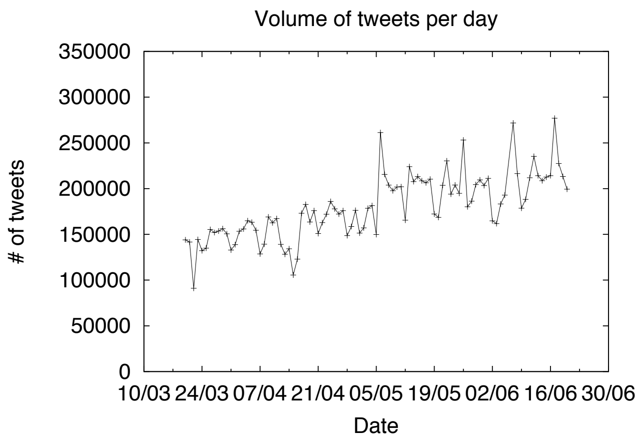

Figure 2: Daily volume of tweets.

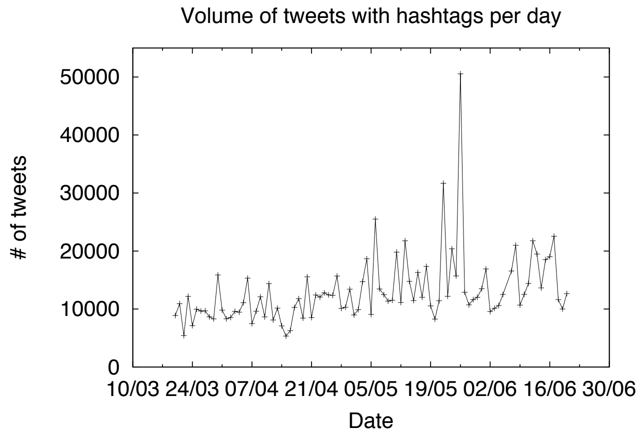

Figure 3: Daily volume of tweets with at least one hashtag.

For the time period during which we performed the data collection, two general elections took place in Greece (on May 6 and June 17) and politics-related discussions are prominent in the dataset we have collected. Indeed, the second most frequent hashtag is #ekloges12 (which stands for elections 2012) with frequency 46552. The most frequent hashtag is #ff with frequency 59892 and the number of distinct hashtags is 163347.

The collected dataset is loaded in a relational database as follows. First, we populate the relation TT with tuples for each hashtag found in the collected tweets and the corresponding four-hour time period during which it occurred. Next, for each timespan of four hours, we generate the model's quintuples and store them in a relation called M. The attributes of relation M correspond to the elements of quintuples and the counts required to compute the association weight, as defined in Equation 1. The inclusion of extra information in the relation M facilitates the efficient computation of association weights without having to process again information from the relation TT. This feature is important in the case of the *fold* and *merge* operators, as we will see in Section 4.2.

In our setting, the time instance $t = 1$ corresponds to March 20, 2012 04:00 GMT. The number of tuples in relations TT and M is 1568673 and 733677, respectively.

4.2 Query processing description

In this section we describe the steps of computing the result for the following request:

> Find the hashtags that are associated with #ekloges12 and for which the association weight increases for two consecutive weeks.

In other words, the above request asks for hashtags which co-occur with hashtag #ekloges12 with increasing probability for a period of two weeks. If we denote the input model as $\mathcal{M}_1$, then we can express the above request as the sequence of operators below:

$$\begin{aligned}
\mathcal{M}_2 &= filter(\mathcal{M}_1, n = \#\text{ekloges12})\\
\mathcal{M}_3 &= fold(\mathcal{M}_2, 42)\\
\mathcal{M}_4 &= join(\mathcal{M}_3 \text{ as } m, \mathcal{M}_3 \text{ as } m', cond)\\
\mathcal{M}_5 &= join(\mathcal{M}_4 \text{ as } m, \mathcal{M}_4 \text{ as } m', cond)
\end{aligned}$$

where the condition *cond* corresponds to:

$$m.n <> m.c \wedge m.n = m'.n \wedge m.c = m'.c \wedge$$
$$m.w > m'.w \wedge min(m.T) = max(m'.T) + 1$$

The condition *cond* is satisfied by quintuples m and m' which have the same n and c nodes, respectively, m has a higher weight than m', the timespan of m immediately follows that of m' and the target and context nodes of m are different.

The first operator *filter* selects from the input model $\mathcal{M}_1$ the quintuples for which the target node is the given hashtag #ekloges12. From the 733677 quintuples in $\mathcal{M}_1$, the filter operator results in $\mathcal{M}_2$ with 8074 quintuples. We implement the *filter* operator using an SQL select query. The operator *fold* creates new quintuples with granularity equal to one week (42 consecutive timespans of four hours). The model $\mathcal{M}_3$ contains 2830 quintuples. The *fold* operator first reads the output of the *filter* operator and merges quintuples which fall in the same timespan of 42 consecutive time instances. For example, all the quintuples for a given pair of n and c and $t \in T$ where $1 \leq t \leq 42$ are merged into one quintuple with $T = \{1 \dots 42\}$ and granularity $g = 42$. The updated weights are computed from the counts stored along with each quintuple in the relation M.

The first *join* operator results in the model $\mathcal{M}_4$ which contains quintuples for which the association weight from n = #ekloges12 to c increases from one week to the next one. Similarly, $\mathcal{M}_5$, which is the model obtained from the application of the second *join* operator, contains the quintuples for which the association weight from n = #ekloges12 to c increases for two consecutive weeks. The *join* operators are implemented using an SQL semi-join with further processing in order to check that the timespans of selected quintuples satisfy the condition *cond*, because the relational database does not support comparisons of sets of integers. Model $\mathcal{M}_5$ has a total of 85 quintuples.

Table 1: The quintuples for which n = #ekloges12 and c = #eklogesgr and the intermediate models which contain them.

Quintuple	Models
$\langle$#ekloges12, #eklogesgr, 0.0048, $\{169 \dots 210\}$, 42$\rangle$	$\mathcal{M}_3$
$\langle$#ekloges12, #eklogesgr, 0.0015, $\{211 \dots 252\}$, 42$\rangle$	$\mathcal{M}_3$
$\langle$#ekloges12, #eklogesgr, 0.0031, $\{253 \dots 294\}$, 42$\rangle$	$\mathcal{M}_3, \mathcal{M}_4$
$\langle$#ekloges12, #eklogesgr, 0.0004, $\{295 \dots 336\}$, 42$\rangle$	$\mathcal{M}_3$
$\langle$#ekloges12, #eklogesgr, 0.0036, $\{337 \dots 378\}$, 42$\rangle$	$\mathcal{M}_3, \mathcal{M}_4$
$\langle$#ekloges12, #eklogesgr, 0.0136, $\{379 \dots 420\}$, 42$\rangle$	$\mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5$
$\langle$#ekloges12, #eklogesgr, 0.0011, $\{421 \dots 462\}$, 42$\rangle$	$\mathcal{M}_3$
$\langle$#ekloges12, #eklogesgr, 0.0032, $\{463 \dots 504\}$, 42$\rangle$	$\mathcal{M}_3, \mathcal{M}_4$
$\langle$#ekloges12, #eklogesgr, 0.0030, $\{505 \dots 546\}$, 42$\rangle$	$\mathcal{M}_3$
$\langle$#ekloges12, #eklogesgr, 0.0010, $\{547 \dots 588\}$, 42$\rangle$	$\mathcal{M}_3$

Table 2: The 10 quintuples from model $\mathcal{M}_5$ with the highest weight.

$\langle$#ekloges12, #pasok,	0.08794, $\{421 \dots 462\}$, 42$\rangle$
$\langle$#ekloges12, #samaras,	0.06469, $\{505 \dots 546\}$, 42$\rangle$
$\langle$#ekloges12, #syriza,	0.04663, $\{463 \dots 504\}$, 42$\rangle$
$\langle$#ekloges12, #ekloges2012,	0.04537, $\{253 \dots 294\}$, 42$\rangle$
$\langle$#ekloges12, #2012ek,	0.02956, $\{463 \dots 504\}$, 42$\rangle$
$\langle$#ekloges12, #cpel2012,	0.02859, $\{379 \dots 420\}$, 42$\rangle$
$\langle$#ekloges12, #ekloges2012,	0.02780, $\{421 \dots 462\}$, 42$\rangle$
$\langle$#ekloges12, #cpel2012,	0.02140, $\{337 \dots 378\}$, 42$\rangle$
$\langle$#ekloges12, #mega,	0.01724, $\{463 \dots 504\}$, 42$\rangle$
$\langle$#ekloges12, #eklogesgr,	0.01361, $\{379 \dots 420\}$, 42$\rangle$

In Table 1 we show an example of quintuples with n = #ekloges12 and c = #eklogesgr from the models generated by the *fold* and *join* operators during the processing of the example request. The first column shows the quintuple and the second column shows the models which contain the quintuple. The 110 quintuples for n = #ekloges12 and c = #eklogesgr in $\mathcal{M}_2$ are folded to 10 quintuples in $\mathcal{M}_3$, as shown in Table 1. The output of the second join returns one quintuple for $T = \{379 \dots 420\}$ with granularity $g = 42$ for n = #ekloges12 and c = #eklogesgr.

Table 2 shows the 10 quintuples from $\mathcal{M}_5$ with the highest weight. In the top three quintuples, #ekloges12 is associated with the names of two political parties and the name of one political party leader. In the remaining seven quintuples, #ekloges12 is associated with other hashtags which are used to denote the topic of general elections in 2012, as well as the hashtag #mega, which is the name of a greek television channel.

5. RELATED WORK

The introduced temporal term association model and the related query language constitute a step towards a framework that supports the large-scale and declarative exploration of data obtained from social networks. Smith and Barash [14] have surveyed visualization tools for social network data and stress the need for a language similar to SQL but adapted to social networks. San Martín and Gutierrez [13] describe a data model and query language for social networks based on RDF and SPARQL. However, they do not directly support different granularities of time. Mustafa *et al.* [10] use Datalog to model social networks and to apply data cleaning and extraction techniques using a declarative language.

The most related work to ours is [4] by Doytsher *et al.*, who connect the social network of users with a spatial network to identify places visited frequently by users. The model and query language in [4] allow to query with different granularities for frequency, time and locations. However, they do not consider any text artifacts generated by users (*e.g.* comments posted on blogs, reviews, tweets, *etc.*).

The support for a *fold* operator relates to temporal databases and in particular query languages that enable switching between equivalent representations that utilize intervals and points for time stamping; Bohlen *et al.* [2] provide an overview of temporal aggregation concepts.

Finally, the calculation of association weights defined in Equation 1 relies on the probability of term co-occurrence in order to estimate the strength of association between two terms. This approach is only one of the possible ones we can apply to estimate the association between terms. For example, Dunning [5] proposed the use of the log-likelihood ratio as a more robust estimate of the association between terms. The definition of the *jump* operator relates to the TextRank algorithm [9], which has been used to extract keywords from texts by applying a PageRank-like algorithm to a graph of terms created from a document. Liu *et al.* [8] have proposed to rank tags from Flickr photos by applying PageRank to the similarity graph of tags.

6. CONCLUSIONS

In this work, we have introduced a model and a set of query operators for exploring the associations between terms from texts and social network data, as well as their temporal evolution. The model captures the associations of varying degrees between terms for different granularities of time. The operators allow to manipulate and enrich the data by changing the granularity of time, selecting subsets of terms, adding new term associations or merging existing ones. Furthermore, they can be combined to form complex queries. We have shown a running example for one query and data collected from Twitter, where the model was built from the hashtags found in tweets.

As far as limitations of the introduced model and operations are concerned, they do not support temporal properties for nodes, for example it is currently not possible to select a node according to the volume of its occurrences at a specific timespan. Moreover, the operators do not directly support the use of user-defined functions for computing the association weights. One example of such a user-defined function is the correlation between the frequency time series of two terms, which do not necessarily co-occur in the same tweets. In order to introduce user-defined functions for computing the association weights, we first need to further investigate what additional data is required for the efficient computation of a range of possible functions.

For future works, we plan to address the limitations discussed above and to provide a full implementation of the model and the operators which can run over a relational database. We also plan to experiment with alternative definitions of term associations, beyond co-occurrence, as well as to test the framework with larger datasets comprising both hashtags as well as plain terms.

7. REFERENCES

[1] S. Asur and B. A. Huberman. Predicting the future with social media. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '10, pages 492–499, 2010.

[2] M. H. Bohlen, J. Gamper, and C. S. Jensen. How would you like to aggregate your temporal data? In *Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning*, TIME '06, pages 121–136, 2006.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, Apr. 1998.

[4] Y. Doytsher, B. Galon, and Y. Kanza. Querying geo-social data by bridging spatial networks and social networks. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*, LBSN '10, pages 39–46, 2010.

[5] T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19(1):61–74, Mar. 1993.

[6] B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Micro-blogging as online word of mouth branding. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, CHI EA '09, pages 3859–3864, 2009.

[7] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 151–160, 2011.

[8] D. Liu, X.-S. Hua, L. Yang, M. Wang, and H.-J. Zhang. Tag ranking. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 351–360, 2009.

[9] R. Mihalcea and P. Tarau. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004.

[10] W. E. Moustafa, G. Namata, A. Deshpande, and L. Getoor. Declarative analysis of noisy information networks. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering Workshops*, ICDEW '11, pages 106–111, 2011.

[11] B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith. From tweets to polls: Linking text sentiment to public opinion time series. In W. W. Cohen and S. Gosling, editors, *ICWSM*, 2010.

[12] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 851–860, 2010.

[13] M. San Martín and C. Gutierrez. Representing, querying and transforming social networks with rdf/sparql. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009 Heraklion, pages 293–307, 2009.

[14] M. A. Smith and V. Barash. Social sql: Tools for exploring social databases. *IEEE Data Eng. Bull.*, 31(2):50–57, 2008.