Assessing Trustworthiness in Social Networks using Run-Time Event Recognition

Christos Vlassopoulos[1,2], Alexander Artikis[3,1] and Georgios Paliouras[1]
[1]Institute of Informatics and Telecommunications, N.C.S.R. "Demokritos", Greece
[2]Department of Informatics and Telecommunications, University of Athens, Greece
[3]Department of Maritime Studies, University of Piraeus, Greece
{cvlas, a.artikis, paliourg}@iit.demokritos.gr

ABSTRACT

We present a system aiming to assess the extent to which a Twitter user's reference to a popular topic can be trusted. This decision is based on several factors such as the influence and reputation of the respective user, and the relevance between the topic they are talking about and their interests. Our system takes a stream of tweets as input and is able to produce its results in real time. In order to quickly handle large amounts of data, our system uses a scalable Composite Event Recognition system based on the Event Calculus.

Keywords

Social Networks; Content Trustworthiness; Composite Event Recognition; Run-Time Event Calculus

1. INTRODUCTION

Social Networks have quickly become an indispensable part of our everyday life. They are used not only for communication and socialization purposes, but also for spreading news and/or rumours, discussing the actuality and opinion exchanging. Since more and more people use Social Networks as their everyday source of information, there is a clear need to examine the posted content, test its validity and estimate how likely it is to talk about or reproduce accurate and reliable information or potentially spreading hoax. In other words, we want to be able to decide whether we can *trust* a reference to a topic of interest without running the risk of being misled.

Twitter is one of the most popular Social Networks for reading news. It is easy to use and its posts (tweets) are short to aid quick reading. In this paper we present a system that uses the Event Calculus and Logic Programming on Twitter stream data, in order to support the decision about the trustworthiness of specific posted content.

This system has been designed to produce accurate results as quickly as possible. Specifically, it is a system that takes as input a stream of tweets and produces its results in real time. To that effect, our system is based on a fast and scalable Composite Event Recognition engine that has been successfully used in similar contexts in the past.

ACM ISBN N/A.
DOI: N/A

2. THE PROBLEM

The main challenge we are tackling in this paper is the efficient mining of knowledge from streaming data that stem from social networks. In the context of the REVEAL project[1], we collected datasets of user activity on Twitter and we were interested in labelling content as trustworthy or not, depending on the social links of the user that provided it and the topic it refers to.

Previous work on the field of trustworthiness in Social Networks has focused on assessing the trustworthiness of a user without distinguishing between the various topics they are referring to. The rationale of our approach is that a Social Network user is more probable to be trustworthy about topics that belong to their interests and they mention them often enough, than for topics that they rarely talk about. This assumption has lead us to constructing a system that estimates trustworthiness by focusing on the *content*, rather than the *user*. Our target is to be able to label content as either worth reading or not. Should we be able to fulfil this task, we could use the results of this approach to a variety of other tasks, such as to generalize our assessment to the level of the user, using some aggregation over the entire content posted by a user or to evaluate overall trustworthiness of the discussions about a popular topic, by aggregating over the references to this topic.

The input data is split into two main types: streaming and static. Streaming data comprise the tweets that occur and are received in a chronologically sorted manner. Static data correspond to the background knowledge that we already have about specific aspects of the user activity under examination. For instance, we may a-priori know the lists that our users belong to and/or the topics that are relevant to these lists. We may also be aware of the influence score of each user. These are examples of background knowledge in this case.

There are several issues that add to the challenge. One such issue is the definition of trustworthiness. We should define trustworthiness in such a way, as to lead us to a quantifiable algorithmic estimate. Another requirement is that our system be able to produce real-time results. The input data will be provided in a streaming fashion and the computations should be performed on-the-fly, yielding results with as little latency as possible.

[1]http://revealproject.eu

3. RUN-TIME EVENT CALCULUS

From the description of the problem above, it follows that our approach must be characterized by expressiveness and speed. We need expressiveness in order to define trustworthiness in an intuitively valid and accurate way, and speed in order to achieve real-time computations.

We chose to use the Run-Time Event Calculus[2] (hereinafter RTEC) [1], in order to deal with this issue, because it combines these two desired qualities; expressiveness and speed. RTEC is a dialect of the Event Calculus [9] designed in such a way, as to perform effective Event Recognition and function in streaming environments. The purpose of RTEC is to recognise Composite Events (CEs); i.e., to extract high-level, non-trivial knowledge from streams of low-level data. The only requirement is that we must describe our CEs of interest in terms of lower-level entities.

3.1 Event Description

The *Event Description* is a set of rules (clauses) that describe what kind of low-level events occur in our world and how they affect its state by triggering composite events (CEs). We construct these clauses using the special predicates of the Event Calculus (see table 1). These predicates are used to describe the occurrence of events, as well as the holding or changing of specific qualities within our world.

An event description includes, but is not limited to, the predicates of table 1. These predicates are meant to express temporal constraints within a rule. However, there can also be other, atemporal constraints which can be expressed via domain-specific, user-defined predicates that constitute the world's background knowledge.

3.2 Events and Fluents

The building blocks of an Event Description are events and fluents. Events are *instantaneous* – they occur at some point in time and they usually cause changes in the properties of the world, according to the Event Description. They can be found inside a `happensAt` predicate. Events may appear in either the head or the body of a rule. In other words, they can be input or output entities, simple or composite ones.

In our system, the events are extracted from the input stream of tweets. Whenever a new tweet arrives, we are interested in isolating the main entities that the author mentions and we construct the event

$$namedEntity(UserId, TweetId, NamedEntity) \quad (1)$$

On the other hand, fluents are durative entities. They have values and usually describe some attribute or state of the world under examination. Fluents are found inside the `initiatedAt`, `terminatedAt`, `holdsAt` and `holdsFor` predicates. They are split into two categories: Simple Fluents and Statically Determined ones.

Simple Fluents are properties that start holding at some point in time and may or may not cease to hold later on. They are defined in the Event Description through initiating and terminating statements, using the `initiatedAt` and `terminatedAt` predicates. The initiation and termination conditions include the happening of one or more events and, potentially, the holding of some fluents at the respective initiation or termination instant. RTEC uses the law of inertia to deduce that the simple fluent continuously holds between its initiation and termination points.

In the case that we are studying, the trustworthiness of a reference can take one of three values: "low", "medium", "high" and for each of these values the fluent is initiated by a different set of conditions. Therefore, we model it using the following simple fluents:

$$\begin{array}{l} trust(UserId, TweetId, NamedEntity) = low \\ trust(UserId, TweetId, NamedEntity) = medium \\ trust(UserId, TweetId, NamedEntity) = high \end{array} \quad (2)$$

Statically Determined Fluents are generally more complex than Simple Fluents. A statically determined fluent is only defined in terms of other fluents and has no initiation or termination conditions. Instead, the intervals during which it holds are a combination of the holding intervals of its constituents via interval manipulation operations. Table 1 provides the three interval manipulation operations that RTEC supports; namely the union, the intersection and the relative complement of constituent intervals. In the event description, a statically determined fluent is defined via the `holdsFor` predicate. In the context of our study, however, no Statically Determined Fluents are used.

We encourage the reader that is interested in finding out more about RTEC to see [1], as well as the user's manual[3].

4. THE ASSESSMENT PROCESS

4.1 Data sources

In section 2 we mentioned the separation of our data into streaming and static. Streaming data usually produces the event narrative of our scenario, whereas static data is used as atemporal, background knowledge. In the context of the REVEAL project, there are four sources of data;

1. A stream of tweets and their meta-data.
2. A list of all users and their influence score.
3. A list that associates users to Twitter lists.
4. A list that matches named entities with the relevant Twitter lists.

These data are the result of earlier preprocessing in the REVEAL system. The first source is used in order to extract the named entities of a tweet and construct the low-level event *namedEntity*(*UserId*, *TweetId*, *NamedEntity*), as shown in section 3.2. In the second source, the user influence is a measure of each user's reputation and impact, and is calculated with the method described in [8]. Its values range between 0 (non-influential) and 1 (very influential), but are discretized into "high" and "low", for ease of understanding. In the third source, the users are related to various Twitter lists, along with a relevance score that is again discretized into the values "high" and "low". This assigning of lists to users is an indication of the users' interests and specializations.

In order to make these "user-list" associations, we first calculate the similarity between tweets and lists, using the

[2]Additional information and source code about RTEC can be found at the RTEC project site, in Github (https://github.com/aartikis/RTEC)

[3]https://github.com/aartikis/RTEC/blob/master/RTEC_manual.pdf

Predicate	Meaning
`happensAt`$(E,\ T)$	Event E occurs at time T
`initially`$(F=V)$	The value of fluent F is V at time 0
`holdsAt`$(F=V,\ T)$	The value of fluent F is V at time T
`holdsFor`$(F=V,\ I)$	I is the list of the maximal intervals for which $F=V$ holds continuously
`initiatedAt`$(F=V,\ T)$	At time T a period of time for which $F=V$ is initiated
`terminatedAt`$(F=V,\ T)$	At time T a period of time for which $F=V$ is terminated
`union_all`$(L,\ I)$	I is the list of maximal intervals produced by the union of the lists of maximal intervals of list L
`intersect_all`$(L,\ I)$	I is the list of maximal intervals produced by the intersection of the lists of maximal intervals of list L
`relative_complement_all`$(I',\ L,\ I)$	I is the list of maximal intervals produced by the relative complement of the list of maximal intervals I' with respect to every list of maximal intervals of list L

Table 1: The predicates of RTEC.

Word2Vec model [12]. A tweet belongs to a list if it contains at least one word that has a Word2Vec similarity above 0.7 with the list[4]. A user is associated with a list if more than 30% of their tweets belong to the list. Finally, our last source of data provides the Twitter lists that are related to each named entity. Again, this association is determined via Word2Vec similarity measurements, this time between the tokens of a named entity and a list. A named entity is considered to belong to a list if at least one of its tokens is similar to this list.

4.2 Methodology

The above input data sources are used as the building blocks of the definition of trustworthiness. Recall that, as we mentioned in section 2, our goal is to assess the trustworthiness of the posted content, i.e. the tweets. Whenever we come accross a tweet, we want to be able to decide whether it can be trusted or not. However, since each tweet contains a group of named entities, we express trustworthiness in terms of each named entity, within a tweet. For instance, if user "chris08" writes a tweet with id "01136" and names the entities "US", "Election", "Vote", we will perform separate trustworthiness estimation for each of the three named entities:

$$\begin{aligned} &trust(chris08, 01136, us) = high/medium/low \\ &trust(chris08, 01136, election) = high/medium/low \\ &trust(chris08, 01136, vote) = high/medium/low \end{aligned} \quad (3)$$

These separate composite events that concern the same tweet can be merged, later on, into a single trustworthiness estimate for the entire tweet. Hence, the question we need to answer is "*What are the conditions that must be satisfied in order for a tweet, posted by a certain user and referring to specific entities, to be trusted?*".

As the assessment is based on named entities, the first step is to identify the entities mentioned within a tweet. Speaking the language of the Event Calculus, the reference to a named entity is an instantaneous event (see condition 1) and stems from the streaming part of the data, and thus its occurrence should be stated with the `happensAt` predicate.

The first trustworthiness criterion that we use is that the topic the tweet talks about is relevant to its author's interests. A good way to see the interests of a user is to study the Twitter lists in which they belong. To that effect, we draw the associations between users and lists from the respective data source discussed in 4.1. This background information is expressed by the following predicate:

$$inList(UserId, ListName, high/low) \quad (4)$$

For instance, if user "chris08" is related to the "politics" list with a high relevance score, then this piece of information would be represented as $inList(chris08, politics, high)$.

A user that is part of a specific list is considered more trustworthy when they talk about a topic that is relevant to this list than otherwise. Hence, we require that the user belong to a list that is of relevance to the named entity they are talking about. This condition is assessed with the data source that provides the associations between named entities and lists, and is expressed as follows:

$$related(NamedEntity, ListName) \quad (5)$$

For example, named entity "Election" being associated with the "politics" list is expressed as $related(election, politics)$.

Further to the relevance of the topic of the tweet to the interests of the user, we use additional evidence about the user, in order to increase or decrease our estimate of trustworthiness. In this work, this evidence comes in the form of user influence. An influential user is considered to be more trustworthy, than one who has little activity in the social network. Hence, we create one final condition, as follows:

$$influence(UserId, high/low) \quad (6)$$

If user "chris08" happens to be very influential, then the condition becomes $influence(chris08, high)$.

[4]This threshold was decided after experimentation

Based on the above conditions, we can use the Event Calculus formalism and define trustworthiness straightforwardly, as shown in Listing 1:

```
initiatedAt(trust(UserId,TweetId,NamedEntity)=high,Time):-
       happensAt(namedEntity(UserId,TweetId,NamedEntity),Time),
       related(NamedEntity,ListName),
       inList(UserId,ListName,high),
       influence(UserId,high).

initiatedAt(trust(UserId,TweetId,NamedEntity)=medium,Time):-
       happensAt(namedEntity(UserId,TweetId,NamedEntity),Time),
       related(NamedEntity,ListName),
       inList(UserId,ListName,low),
       influence(UserId,high).

initiatedAt(trust(UserId,TweetId,NamedEntity)=medium,Time):-
       happensAt(namedEntity(UserId,TweetId,NamedEntity),Time),
       related(NamedEntity,ListName),
       inList(UserId,ListName,high),
       influence(UserId,low).

initiatedAt(trust(UserId,TweetId,NamedEntity)=low,Time):-
       happensAt(namedEntity(UserId,TweetId,NamedEntity),Time),
       related(NamedEntity,ListName),
       inList(UserId,ListName,low),
       influence(UserId,low).
```

Listing 1: The complete definition of trustworthiness

The code in Listing 1 shows four different trustworthiness scenarios. In the first scenario, the user is influential, the topic (entity) they talk about is strongly related to their interests and therefore the mention is considered to be highly trustworthy. In the second scenario the user is influential, but the topic they talk about is weakly related to their interests. Therefore the trustworthiness estimate decreases to medium. The same score is given to the situation where the user is not influential, but there is a strong relevance between the topic and their interests. Thus the user is considered again of medium trustworthiness. Finally, in the fourth scenario, the user is neither influential, nor is there a significant relevance between their topic and their interests. Hence, the content is considered of low trustworthiness.

From the discussion above, it becomes clear that the definitions are simple, in the language of RTEC. Using only four conditions, we managed to combine the topics that the users refers to, their interests and their influence, and defined four trustworthiness scenarios. If new data become available and thus more qualities and attributes can be specified (e.g.: the popularity of a topic or tweet), then we can simply append them to the already existing rules to form bigger and more sophisticated ones. In addition, the existence of more conditions and characteristics in the definition of trustworthiness will lead to even more possible combinations and scenarios concerning the trust levels.

4.3 Implementation

Our application draws its input data and background information from the data sources discussed earlier. There is a preprocessing stage, where the text data is cleaned from characters that fall outside the alphanumeric spectrum, and then the data is used to extract the events and conditions, as discussed earlier in this section. Finally, these events and conditions are read by RTEC, during the Event Recognition task, and the recognized composite events are produced and stored in a database.

RTEC is designed to avoid computational overhead. A key implementation technique that helps in boosting RTEC's performance is the "Dynamic grounding". For the sake of completeness, it is important to note here that the *grounding* of an atom of the form $pred(Var1, Var2)$, where $pred$ is a predicate and $Var1$, $Var2$ are the variables associated with this predicate, is the process of substituting the variables with actual values. For instance, $trust(chris08, 01136, election) = high$ is a possible grounding of the atomic formula $trust(UserId, TweetId, NamedEntity) = high$.

As the low-level events (see statement (1)) stream into our system, RTEC stores the values of the arguments within the low-level events. In our case, the arguments of the low-level events are the id of the user, the id of the tweet and the named entity. These three values will be used later on for the grounding of the composite event $trust(UserId, TweetId, NamedEntity) = high/medium/low$. The stored values from a *grounding domain* that grows, as each new low-level event produces a new grounding domain entry.

Whenever RTEC attempts to recognize a composite event (the trustworthiness in this case), it scans the grounding domain and for each grounding tuple (UserId, TweetId, NamedEntity), it grounds the *trust* formula using these values, thus forming a new composite event, and tries to find holding intervals for this composite event. If RTEC finds holding intervals for the composite event, this means that it has found trustworthy content and the composite event is stored and shown on the system's output. Otherwise, it produces no output and moves on to the next grounding tuple.

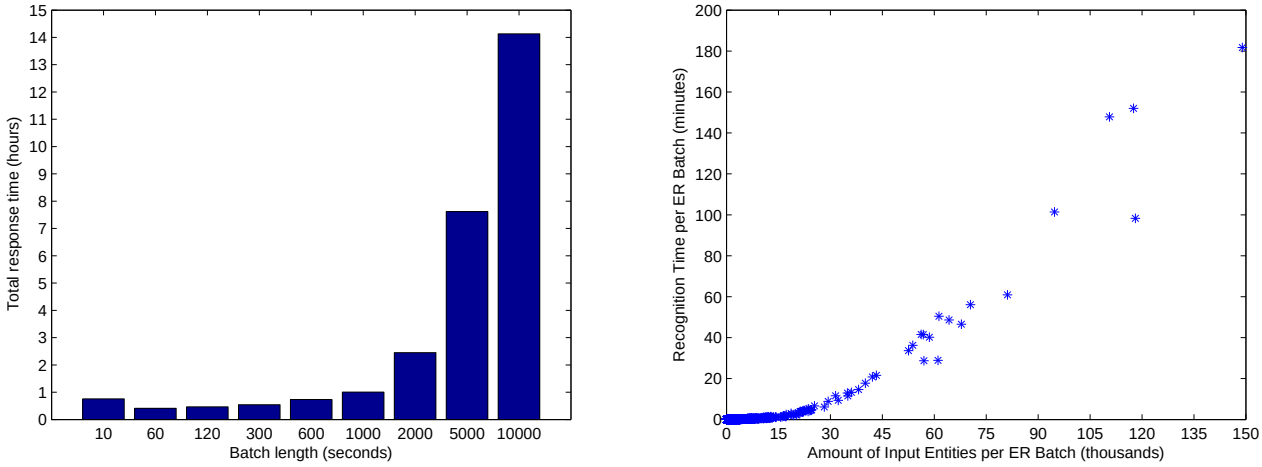
This iteration over the grounding tuples results in significant delays in the response time, as the grounding domain grows. Therefore, it is crucial that we keep the grounding domain as short as possible. RTEC's "Dynamic grounding" takes care of this issue and, after each Event Recognition step, all the grounding tuples are discarded, except for those that correspond to composite events that have started holding, but their termination point has not yet been found by RTEC. This means that we discard the grounding tuples that do not participate in any composite event, as well as those that participate in composite events that have already ceased holding. We only keep the grounding tuples that correspond to incomplete composite events, for which the recognition procedure will continue in the next steps, until a termination point is found (for more details, see [1]).

As a result, the dynamic grounding of RTEC prunes the grounding domain periodically, thus accelerating the event recognition process.

5. EXPERIMENTAL RESULTS

The input data are stored in a Mongo database that contains 4 collections, one for each data source presented in section 4.1. Specifically, the "named entities" collection contains 937903 entries (tweets) that are posted during a timeline that spans 24 hours, 50 minutes and 15 seconds (89415 seconds). The "user influence" collection contains information about how influential each one of 503228 users is, whereas the "users-lists" collection hosts the classification of 577611 users into Twitter lists, and "named entities-lists" shows for each of 100932 named entities their relevant lists.

For the purposes of our experiments, the dataset is broken into smaller batches and, for each batch, high-level events of interest are recognized based on the provided definitions. To test the performance of the system, we tuned the amount of data it receives as input, in each batch. By increasing the



(a) Total experiment duration

(b) ER duration over Low-level events

Figure 1: The relationship between the system's response time and the size of the input

Batch length	Number of batches	Total ER duration (minutes)	ER duration per batch (seconds)
10	8942	45.3	**0.3**
60	1491	**24.78**	0.99
120	746	27.8	2.23
300	299	32.62	6.55
600	150	44.03	17.61
1000	90	60.39	40.26
2000	45	146.96	195.94
5000	18	457.25	1524.2
10000	9	847.86	5652.4

Table 2: Summary of the total and average response times

length of the batches, we increase the amount of input data at each step of our system's Event Recognition process. We tried 9 different batch lengths, as it can be seen in table 2.

In the first experiment, we took our 89415-second-long tweet timeline and split it into 8942 batches of length 10. For each of these batches we performed Composite Event Recognition using RTEC. All 8942 tasks took about 45.3 minutes to complete, which corresponds to an average of 0.3 seconds per batch. In the second experiment, we increased the batch length to 60 seconds. Our system performed Composite Event Recognition for a total of 1491 batches in 24.78 minutes, averaging 0.99 seconds per batch. Doubling the previous batch length resulted in a total response time of 27.8 minutes (2.23 seconds per batch, on average). Similarly, increasing our system's batch length parameter resulted in longer execution times, both on average and in total. Table 2 contains all duration measurements and Figure 1a gives a graphical representation of the total execution times.

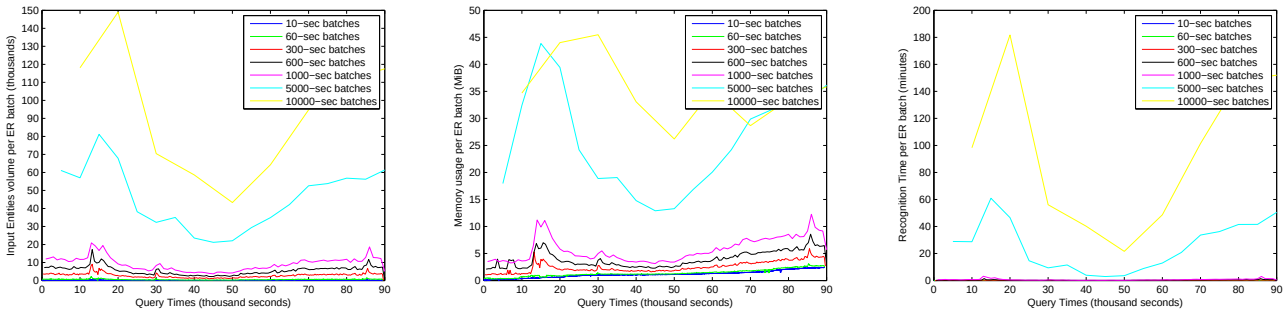
It can be inferred from Table 2 and Figure 1a that tiny batches of input data result in extremely reduced average Event Recognition duration. In total, however, the setting that yields the quickest completion is not the one with the smallest batch length. This is because the performance of the system is also affected by the number of steps required in order for the whole data stream to be processed. Setting the batch length equal to 60 may triple the average processing time per step, but, at the same time, it reduces the number of steps to just $\frac{1}{6}$ of the 10-second-long batch setting. This results in a much quicker response, overall.

We should also point out that based on these performance measurements, whatever the step size might be, the system is always real-time, as the average response time per step is significantly lower than the step size itself. The average response time, however, shows signs of non-linear increase as the step size increases. An increase in the Event Recognition step causes more input data in the form of low-level events each time. Figure 1b illustrates the effect of the input data volume to the Event Recognition step duration. It is evident that there is non-linear, increasing trend over the input volume. Nevertheless, even if we set a batch length equal to 10000 seconds, which means that there are tens or hundreds of thousands of input entities per step (see also Figure 2a) the system remains real-time, as the average processing time per step does not exceed the actual timespan of the data within a batch.

Figure 2a shows the input entity volume, for each of the experiments[5] conducted. In all experiments with step sizes 1000 seconds or less, the system received no more than 20000 input entities in a single batch. On the other hand, if we increase the batch length to 5000 or 10000 seconds, the system needs to deal with 50000, 100000 or even 150000 input entities per batch.

The input data volume directly affects the memory usage. As the system receives low-level events, it stores them and at the same time it produces grounding tuples for the output entities, thus enriching the grounding domain. In Figure 2b we can see that the memory usage per step for each experimental setting follows a similar trend to the input data volume. Batch lengths of up to 1000 seconds use 12MiB or

[5]For readability and presentability reasons we exclude the 120 and 2000-second-long batch settings



(a) The input entities (b) Memory consumption (c) ER duration

Figure 2: The workload of our system in terms of low-level event volume, memory consumption and response time per event recognition step

less, whereas bigger batches appear to cause the system to consume up to 45MiB of memory.

Another feature that seems to behave according to the input data volume is the Event Recognition duration. Figure 2c suggests that in the experimental settings with batch lengths of up to 1000 seconds, the Event Recognition duration does not exceed 3 minutes. However, for batch lengths of 5000 and 10000 seconds we observe that the processing times for a single bunch of data can rise up to 3 hours.

6. RELATED WORK

This work has been carried out with a view to contributing to the field of Composite Event Recognition on Social Networks. We should note here, however, that this should not be confused with Natural Language Processing, which has been extensively discussed in the context of Social Networks. Instead, it is an effort to define and detect patterns within the Social Network content and its authors that will help us handle the vast content more effectively, by filtering out rumours/hoax and other unsolicited content and sources.

The topic of trustworthiness in social networks has been widely discussed over the past years, and is still a popular research topic, as the importance of social networks as a means of communication and content sharing is steadily increasing. Modelling and evaluating trust in social networks is a demanding task; It involves the precise definition of the notion of trustworthiness, as well as the quick detection of trustworthy content.

A significant amount of work, as in [11], [3], [14], or [10] has been put to evaluate trust between users that relate to each other in the real world, via friendships, relationships etc.. Strong actual ties between social network users are considered to imply the existence of trust links. However, our approach does not take into account any relationship between Twitter users. On Twitter, it is very common for a user to come across Tweets written by people they do not know in person, or they do not follow, at all. Defining trust using family ties, relationships and acquaintances would result in labelling content as untrustworthy only because it comes from strangers and trustworthy only because it is written by relatives. We are interested in labelling content as trustworthy or not, regardless of the relationship between users.

In [4], [6], [5] and [15] there have been efforts to evaluate trust for users that do not know each other and use this trust estimation to filter Social Network content. This approach resembles ours, in the sense that it studies trust between strangers. Nevertheless, Golbeck et al. also involve the notion of provenance, along with trustworthiness, in their social network content filtering efforts, whereas we, for the sake of simplicity, chose not to include provenance or location information in our experiments. However, future work on our subject involves using such additional information and building more composite and precise trustworthiness definitions.

We should note at this point that the above mentioned approaches mainly aim at defining and detecting trust towards another person, in general. We, on the other hand, define trustworthiness in a more content-based way; We take into account not only the author of a tweet, but also the topic they are tweeting about, among other features and metadata. In other words, we do not blindly trust someone. We evaluate their trustworthiness with respect to what they are talking about. We designed our trustworthiness model so as to allow for different trustworthiness levels for the various topics that may be discussed by a certain use. We consider that when a user tweets about topics that lie within their areas of interest or expertise, then they are more likely to produce trustworthy content.

Other work on content-based trustworthiness evaluation for social networks includes [13]. The authors of this paper have tried to evaluate content trustworthiness by assigning each content unit a trustworthiness score. In their work, trustworthiness is analysed into a group of sub-factors, such as source competence, recency, provenance/proximity, and quality. Each of these sub-factors is assigned a score. Then the scores of all these sub-factors are combined to form a total trustworthiness score for a specific content unit. This work resembles our work in that the trustworthiness evaluation is content-based and that we combine other, low-level conditions, in order to assess the high-level quantity of interest. Nonetheless, in our approach, we declared three levels of trust (low, medium, high), instead of numerical values, and involved the notion of user influence as one of the building blocks of trustworthiness, along with information about the lists that the users belong to and the topics that they talk about. Last but not least, we combined this information using the Event Calculus, whereas Nurse et al. used combiner functions such as the weighted arithmetic mean.

In addition, in [2], Castillo et al. also try to assess content credibility using several factors, namely the reactions that are caused by the content under examination, the popularity of the author, and the the existence of citations. The authors of [2] treated trustworthiness detection as a classification problem, and thus applied Machine Learning methods, such as Bayesian classifiers and Logistic Regression, in their work.

Another research field that is closely related to this work is the field of Social Network analytics. This area includes the retrieval of popular posts, the detection of user communities that are involved in a topic, and the classification of users into influential or not, among others. The ACM's DEBS 2016 Grand Challenge[6] has been focused on the field of Social Network analytics. The participants to this challenge were asked to provide a system capable of discovering popular posts and user communities effectively and in real time, taking streams of text data as input [7]. The proceedings of this conference can be found in the ACM's digital library[7] and contain an abundance of articles that deal with this subject.

7. CONCLUSIONS

The main idea behind the methodology is rather simple. First of all we try to keep our target clear. In this case our target is the recognition of trustworthiness in Twitter content. Then, we try to keep its definition simple. The Event Calculus is a logic formalism that allows for simple definitions even for quite complex notions, since it breaks them down into events, properties and background knowledge.

This approach combines the expressive power of the Event Calculus with the scalability of RTEC and offers us the flexibility to try to recognize any composite notion about Social Networks that can be modeled using events, fluents and background knowledge. Trustworthiness in the Twitter content as described in this paper is just one of many possible composite events that could be defined. We could also define the same composite event in a different way, taking other data sources, attributes and values into account.

The experimental evaluation indicates that this system is able to recognize a Twitter user's trustworthiness in real time. It has been tested using a dataset containing roughly one million actual Tweets in the English language, collected within a 24.8 hour stint (from Tuesday, 25 February 2014, 17:32:20 until Wednesday, 26 February 2014, 18:22:34).

The total response time of our system highly depends on the volume of the input low-level events. The more low-level events rush into the system, the bigger the knowledge base becomes, as each new event causes a new fact to be asserted. RTEC looks inside its knowledge base and tries all the possible groundings for its output entities at each Event Recognition step.

This system can be used in a variety of similar datasets and, depending on how dense they are, by tuning the batch size, we can find an optimal response time for our system.

This work on content trustworthiness in social media can be further expanded in a variety of ways. One could distinguish 3 development axes. One such axis is the refinement of the definition of trustworthiness. More data can be used, more attributes and data sources can be taken into account when building the rules. Thus, there can be more sophisticated definitions that better simulate the actual notion of trustworthiness in social media. Another development axis concerns the definition and recognition of different composite event patterns. Apart from trustworthiness, there are many other interesting aspects of the social networks that can be modeled similarly. For instance, we may also be interested in detecting trends and topics of rapidly increasing or decreasing popularity. Finally, this system could be upgraded in terms of performance with the use of new technologies and implementations. For example, it would be interesting to implement a parallelized and/or distributed Composite Event Recognition mechanism, for even better scalability.

[6]https://debs2016.org

[7]http://dl.acm.org/citation.cfm?id=2933519

8. ACKNOWLEDGMENTS

The authors would like to acknowledge partial support of this work from the European Community Seventh Framework Programme, as part of the FP7 610928 REVEAL (REVEALing hidden concepts in Social Media) project.

Moreover, we would like to thank our colleagues Anastasia Krithara, Dimitrios Vogiatzis, Konstantinos Panousis, Konstantinos Bougiatiotis, Georgios Katsibras and Efstratios Tzoannos for their valuable contribution in constructing the definitions for trustworthiness, in providing the necessary data and integrating this system with the rest of the modules of the REVEAL project.

9. REFERENCES

[1] A. Artikis, M. J. Sergot, and G. Paliouras. An event calculus for event recognition. *IEEE Trans. Knowl. Data Eng.*, 27(4):895–908, 2015.

[2] C. Castillo, M. Mendoza, and B. Poblete. Predicting information credibility in time-sensitive social media. *Internet Research*, 23(5):560–588, 2013.

[3] E. Gilbert and K. Karahalios. Predicting tie strength with social media. In D. R. O. Jr., R. B. Arthur, K. Hinckley, M. R. Morris, S. E. Hudson, and S. Greenberg, editors, *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009*, pages 211–220. ACM, 2009.

[4] J. Golbeck. Combining provenance with trust in social networks for semantic web content filtering. In L. Moreau and I. T. Foster, editors, *Provenance and Annotation of Data, International Provenance and Annotation Workshop, IPAW 2006, Chicago, IL, USA, May 3-5, 2006, Revised Selected Papers*, volume 4145 of *Lecture Notes in Computer Science*, pages 101–108. Springer, 2006.

[5] J. Golbeck. Computing with trust: Definition, properties, and algorithms. In *Second International Conference on Security and Privacy in Communication Networks and the Workshops, SecureComm 2006, Baltimore, MD, Aug. 28 2006 - September 1, 2006*, pages 1–7. IEEE, 2006.

[6] J. Golbeck and A. Mannes. Using trust and provenance for content filtering on the semantic web. In T. Finin, L. Kagal, and D. Olmedilla, editors, *Proceedings of the WWW'06 Workshop on Models of Trust for the Web (MTW'06), Edinburgh, Scotland, UK, May 22, 2006*, volume 190 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[7] V. Gulisano, Z. Jerzak, S. Voulgaris, and H. Ziekow. The debs 2016 grand challenge. *DEBS 2016 proceedings*, June 2016.

[8] G. Katsimpras, D. Vogiatzis, and G. Paliouras. Determining influential users with supervised random walks. In A. Gangemi, S. Leonardi, and A. Panconesi, editors, *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*, pages 787–792. ACM, 2015.

[9] R. A. Kowalski and M. J. Sergot. A logic-based calculus of events. *New Generation Comput.*, 4(1):67–95, 1986.

[10] D. Z. Levin and R. Cross. The strength of weak ties you can trust: The mediating role of trust in effective knowledge transfer. *Management Science*, 50(11):1477–1490, 2004.

[11] P. Mika. *Social Networks and the Semantic Web*, volume 5 of *Semantic Web and Beyond: Computing for Human Experience*. Springer, 2007.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[13] J. R. Nurse, I. Agrafiotis, M. Goldsmith, S. Creese, and K. Lamberts. Two sides of the coin: measuring and communicating the trustworthiness of online information. *Journal of Trust Management*, 1(1):5, 2014.

[14] R. Xiang, J. Neville, and M. Rogati. Modeling relationship strength in online social networks. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 981–990. ACM, 2010.

[15] C. Ziegler and J. Golbeck. Models for trust inference in social networks. In D. Król, D. Fay, and B. Gabrys, editors, *Propagation Phenomena in Real World Networks*, volume 85 of *Intelligent Systems Reference Library*, pages 53–89. Springer, 2015.