`ArmaTweet`: Detecting Events by Semantic Tweet Analysis

Alberto Tonon[1], Philippe Cudré-Mauroux[1], Albert Blarer[2], Vincent Lenders[2], and Boris Motik[3]

[1] eXascale Infolab, University of Fribourg, Switzerland
[2] armasuisse, Switzerland
[3] University of Oxford, United Kingdom

Abstract. armasuisse Science and Technology, the R&D agency for the Swiss Armed Forces, is developing a Social Media Analysis (SMA) system to help detect events such as natural disasters and terrorist activity by analysing Twitter posts. The system currently supports only keyword search, which cannot identify complex events such as 'politician dying' or 'militia terror act' since the keywords that correctly identify such events are typically unknown. In this paper we present `ArmaTweet`, an extension of SMA developed in a collaboration between armasuisse and the Universities of Fribourg and Oxford that supports *semantic event detection*. Our system extracts a structured representation from the tweets' text using NLP technology, which it then integrates with DBpedia and WordNet in an RDF knowledge graph. Security analysts can thus describe the events of interest precisely and declaratively using SPARQL queries over the graph. Our experiments show that `ArmaTweet` can detect many complex events that cannot be detected by keywords alone.

1 Introduction

Twitter[4] is a popular microblogging service. As of late 2016, an estimated 317 million users produce around 500 million messages (or *tweets*) per day that are broadcast to the users' *followers*. Tweets contain up to 140 characters and cover almost any topic, including personal messages and opinions, celebrity gossip, entertainment, news, and more. Current events are widely discussed on Twitter; for example, around 1.7 M tweets were sent on 7/1/2015 in response to the Charlie Hebdo attacks in Paris. Twitter users often provide live updates in critical situations; for example, users tweeted 'In Brussels Airport. Been evacuated afer [sic] suspected bomb.' and 'Stampede now. Everyone running' during the attack at the Brussels airport on 22/3/2016. Most tweets can be read by unregistered users, so Twitter can potentially provide a real-time source of information for detecting newsworthy events before the conventional broadcast media channels. Thus, the development of techniques for tweet analysis and event detection has attracted considerable attention. The Natural Language Processing (NLP) community adapted their techniques to tweets [17, 23, 9], which are short and often

[4] http://twitter.com/

use a colloquial style with nonstandard acronyms, slang, and typos. These tools were used to develop numerous approaches to event detection on Twitter, and we survey the NLP tools and the event detection approaches in Section 2.

Based on these results, armasuisse Science and Technology—the R&D agency of the Swiss Armed Forces—is developing a Social Media Analysis (SMA) system, which aims to help security analysts detect security-related events. Similarly to previous work [2, 15], analysts currently describe the relevant events using keywords, which are evaluated over tweets using standard Information Retrieval (IR) techniques. This approach, however, cannot detect events with complex descriptions. For example, to detect deaths of politicians, an analyst might query the SME system using keywords 'politician die', but this results in both low precision and low recall. For example, the system misses the death of Edward Brooke (the first African American US senator) since, instead of the word 'politician', most tweets about this event contain phrases such as 'the senator' or 'elected to the US Senate'; similarly, the word 'die' is very frequent on Twitter and so the query retrieves mostly irrelevant tweets. To reliably detect such events, one must understand the intended meaning of the query, know which people are politicians, and identify tweets that mention such a person as a subject of a verb 'to die'. Similarly, to match a query for 'militia terror act' to an attack of Boko Haram on a village in Nigeria, one must know that Boko Haram is a militia group and that terror acts include kidnappings and bombings.

In this paper we present `ArmaTweet`—an extension of SMA to *semantic event detection* developed in a collaboration between armasuisse and the Universities of Fribourg and Oxford. Our system uses NLP technique to extract a structured representation from tweets and integrate it with DBpedia and WordNet in an RDF knowledge graph. Users can thus describe relevant event categories by using *semantic queries* over the knowledge graph. The system evaluates these queries using semantic technologies to retrieve the relevant tweets and passes them to an anomaly detection algorithm to determine whether and how they correspond to actual events. We evaluated our system on the 1%-sample of tweets collected by the Twitter's streaming API during the first six months of 2015. The system detected a total of 941 events across seven different event categories. We evaluated our results using three different definitions of which tweets should be considered relevant to the query. Depending on the selected relevance metric, our system achieved precision between 46% and 67% across all categories. Most of these events could not be detected by the previous version of the system, showing clearly how our approach complements standard keyword search.

2 Related Work

Although analysing tweets is very challenging, initiatives such as the Named Entity rEcognition and Linking (NEEL) Challenge have spurred on the NLP community to develop a comprehensive set of tools including Part-of-Speech (POS) taggers [17], Named Entity Recognisers [22, 5], and dependency parsers [9]. To understand the syntactic structure of tweets, our system must identify

dependencies between terms (e.g., identify the subject of a given verb, determine grammatical cases, and so on). We do not know of a Twitter-specific system that provides such functionality, so decided to use the Stanford CoreNLP library [14] that was originally designed to analyse cleaner text.

A recent survey of the methods for event detection on Twitter [7] classifies existing approaches into three groups. The first one contains approaches for detecting *unspecified events*—that is, events of general interest with no advance description. These approaches typically detect trends in features extracted from tweets and/or cluster tweets based on their topic [3,13,29]. Several systems detect breaking news [26, 19, 18], and one additionally classifies events into predefined types such as 'Sports', 'Death', or 'Fashion' [23]. Some approaches use probabilistic similarity instead of clustering [31]. Analogously to these approaches, we also identify events by detecting trends, but only after semantic queries have been used to identify the tweets matching the user's interests (see Section 3).

The second group contains approaches for detecting *predetermined events*, such as concerts [4], controversial events [20], local festivals [10], earthquakes [25], crime and disaster events [12], and disease progression [27]. Such systems are specifically tailored to an event type, and they usually involve training a classifier on manually annotated tweets to learn the correlation of features that identifies tweets talking about an event. The EMBERS system [21] goes a step further by aggregating many sources of information (Twitter, Web searches, news, blogs, Internet traffic, and so on) to detect and *predict* instances of civil unrest.

The third group contains approaches for detecting *specific events*, which typically use IR methods to match a query (i.e., a Boolean combination of keywords) to a database of tweets. Queries are either provided by the users or are learned from the context [2], and recall can be improved by query expansion [15]. These techniques have been combined with geographical proximity analysis to detect civil unrest [30] and model events in Twitter streams [8]. `ArmaTweet` also identifies tweets using queries provided by users and thus, broadly speaking, falls into this category; however, instead of keyword queries, it uses semantic queries describing the relationships between entities in tweets. The system thus supports queries for specific events (e.g., 'Obama meets Trump') that can be captured using keywords, as well as more complex queries specifying an event *type* (e.g., 'somebody hacks a company') for which a keyword-based approach is not effective. Our system does not rely on a training phase, but requires users to specify their interests precisely by constructing semantic queries. An approach most similar to ours constructs a knowledge graph of events from news articles [24], and the main difference to our work is that it focuses on longer, cleaner texts.

3 Motivation & Methodology

Motivation. Detecting Twitter events using complex descriptions (e.g., based on entities' classes or their relationships) is still very challenging. Consider, for example, the 'politician dying' description from the introduction, and the death of Edward Brooke on 3/1/2015. The event has been widely discussed on Twitter,

and running the keyword query 'edward brooke' for that day in SMA returns 121 tweets. This, however, is just a tiny fraction of all tweets produced on that day, and so this event is unlikely to be detected by the techniques for unspecified events (see Section 2); for example, the technique by Ritter et al. [23] detected just five completely unrelated events on that day.[5] Moreover, there are no obvious keyword queries: 'die' returns 5161 mostly irrelevant tweets in SMA, 'politician' returns 46 irrelevant tweets, and 'politician die' and 'senator die' return no results (note that SMA uses just 1% of all tweets). Thus, although 'edward brooke' is an effective query, it is unclear how to construct it from description 'politician dying'. Similarly, it is unclear how to exploit classification-based techniques since common features, such as n-grams or bags of words, are unlikely to reflect the semantic information that Edward Brooke was a politician. Other examples of complex events that we consider in this paper include 'politician visits a country', 'militia terror act', or 'capital punishment by country'.

Approach. Since the objective of armasuisse was to detect events with complex descriptions, we depart from statistical and IR approaches and use semantic search instead. In particular, we use natural language processing to associate each tweet with a set of *quads* of the form (*subject*, *predicate*, *object*, *location*), describing who did what to whom and where; any of these components can be empty, which we denote by $\times$. We also associate with each tweet a set of *entities* whose role (subject or object) in the tweet could not be determined. Subjects, objects, locations, and entities are matched to DBpedia [11], a knowledge base extracted from Wikipedia, and predicates are matched to verb synsets in WordNet [16], an extensive lexicon. Thus, DBpedia and WordNet provide us with a vocabulary and background knowledge for describing complex events. For example, tweets about the death of Edward Brooke are associated with quads of the form `(dbr:Edward_Brooke,wnr:200359085-v,`$\times$`,`$\times$`)`, where `wnr:200359085-v` identifies the synset 'to die' in WordNet, and DBpedia classifies `dbr:Edward_Brooke` as an instance of `yago:Politician110451263`. Our simple quad model cannot represent semantic relationships such as appositions, adverbs, dependent clauses, modalities, or causality. While such relationships would clearly be useful, our evaluation (see Section 7) demonstrates that our model is sufficient for detecting many kinds of complex event that cannot be detected using keywords only.

Semantic Event Descriptions. To use `ArmaTweet`, users must first describe formally the events of interest. To facilitate that, the system provides an intuitive and declarative query interface that allows users to query quads in our knowledge graph while exploiting the background knowledge from DBpedia and WordNet. For example, 'politician dying' events can be precisely described by a query that identifies all quads in our knowledge graph whose subject is of type `yago:Politician110451263`, and whose predicate is `wnr:200359085-v`. As we discuss in Section 5, such queries are matched to the knowledge graph in a way that attempts to compensate for the imprecision of natural language analysis. Queries are currently constructed manually, which allows users to precisely de-

[5] http://statuscalendar.com/month/201501/ accessed on 14 December 2016.

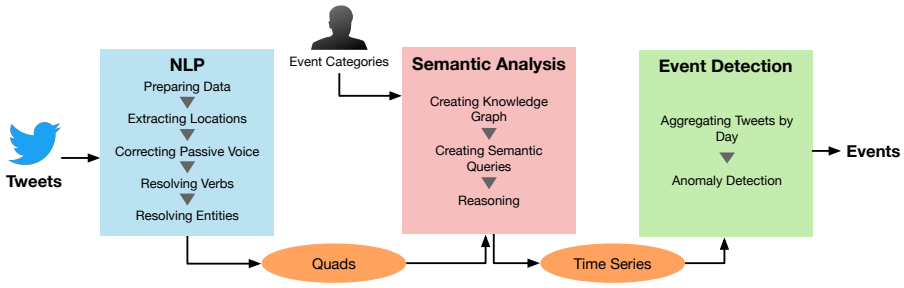

Fig. 1. `ArmaTweet` Architecture

scribe their information needs. In our future work we shall investigate techniques that can automate, or at least provide some help with, query construction.

System Output. Given a set of tweets and a set of queries describing complex events, `ArmaTweet` produces a list of events, each consisting of an event date, an event summary, and a set of relevant tweets. The event summary is specific to the event type; for example, for 'politician dying', it identifies the politician in question, and for 'militia terror act', it identifies the militia group and the verb describing the act. Finally, the set of relevant tweets allows the user to validate the system's output, gain additional information, and possibly initiate an appropriate event response. The system currently does not detect long-running events (e.g., political turmoil or health crises)—that is, each event is associated with a single day only. Thus, the same real-world event can be reported as several events having the same summary but occurring on distinct days. Longer-running events are often reported as events with the same summary occurring in close succession, and we shall investigate ways to exploit this in our future work.

System Architecture. Figure 1 shows the architecture of `ArmaTweet` and its three main components. The *Natural Language Processing* component analyses the tweets' text and extracts the quads and entities, and it is independent of the complex event descriptions. The *Semantic Analysis* component converts the output of the NLP step into RDF, which is then analysed and filtered using the user's event descriptions. The output of this component is a set of *tweet time series*, each consisting of a summary and a set of tweets. Finally, the *Event Detection* component uses an anomaly detection algorithm to extract from each time series zero or more dates that correspond to the actual events. The resulting events and their summaries are finally reported to the user.

To understand the conceptual difference between time series and events, consider the 'militia terror act' event query. Our Semantic Analysis component produces one time series with subject 'Boko Haram' and predicate 'to attack', which contains all tweets talking about attacks by Boko Haram regardless of the time of the tweets. Next, the Event Detection component groups the tweets by time and detects anomalies (e.g., abrupt changes in the number of tweets per day). Since

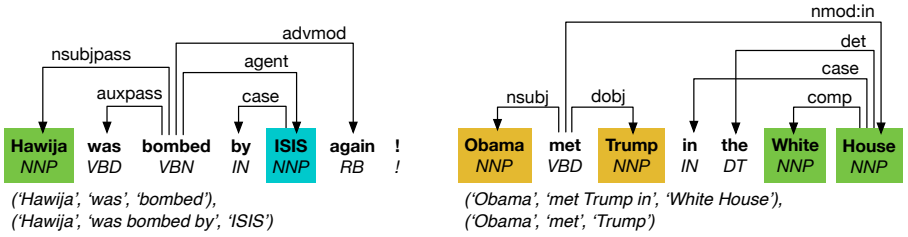

Fig. 2. Data Structures Produced by OpenIE on Two Example Tweets

Boko Haram committed several attacks in our test period, our system extracts and reports several events from this particular tweet time series.

Our NLP processing is computationally intensive, but it is massively parallel since each tweet can be processed independently; hence, we parallelised it using Apache Spark. Moreover, we used the state of the art semantic store RDFox[6] to manage and process our knowledge graph. The parts of our system that are independent from the Spark environment (i.e., the core of the NLP component and the queries/rules used for semantic analysis) are available online.[7]

4 Natural Language Processing of Tweets

The NLP component of `ArmaTweet` extracts from tweets in English a set of quads consisting of a subject, predicate, object, and location, and a set of entities that cannot be assigned to a quad. Predicates are matched to verb synsets in WordNet, and the remaining components are matched to DBpedia resources.

Data Preparation. For each tweet, we first prepare certain data structures. Specifically, we first clean the text by removing emoticons and uncommon characters, we substitute # and @ characters with whitespace, and we split CamelCase words. Next, the OpenIE annotator from the Stanford CoreNLP library [1] transforms the text into *text triples* consisting of a subject, a predicate, and an object; the name 'text triples' emphasises that the components are pieces of text, and not DBpedia or WordNet resources. OpenIE also annotates the mentions of *named entities* (i.e., objects with a proper name) with the entity types (location, organisation, or person); it annotates the text with *part-of-speech (POS)* tags, which describe the relation of a word with adjacent or related words; and it produces a *(dependency-based) parse tree*, which represents the syntactic dependencies between sentence parts using labelled edges between words.

Figure 2 shows the output of OpenIE on two example tweets. The tweet text is shown in bold. Named entity types are coded using colours: the locations 'Hawija' and 'White House' are shown in green, the organisation 'ISIS' is shown

[6] http://www.cs.ox.ac.uk/isg/tools/RDFox/
[7] http://github.com/eXascaleInfolab/2016-armatweet.

in blue, and the persons 'Obama' and 'Trump' are shown in yellow. The POS tags are shown in italic below the words: 'Hawija' is a singular proper noun (*NNP*), 'was' is a verb in past tense (*VBD*), and 'again' is an adverb (*RB*). Finally, the parse trees are shown as labelled arrows connecting words. The roots of the trees are words without incoming edges—'bombed' and 'met' in this case. Moreover, in the rightmost tree, 'Obama' is the subject of the verb 'met' (denoted by a *nsubj* dependency), while 'Trump' is its direct object (denoted by a *dobj* dependency). Finally, the text triples are shown at the bottom of the figure.

Our NLP component also passes the text to DBpedia Spotlight [6], which identifies entity mentions in the text and associates with each mention an appropriate DBpedia resource. For example, on the example shown in Figure 2, Spotlight annotates 'Hawija' with `dbr:Hawija`, 'ISIS' with `dbr:ISIS`,[8] 'Obama' with `dbr:Barack_Obama`, 'Trump' with `dbr:Donald_Trump`, and 'White House' with `dbr:White_House`. We chose Spotlight due to its scalability and ease of use. Spotlight is parameterised by a confidence value that regulates the precision of annotation, and a support value used to filter out uncommon entities, and we empirically determined 0.5 and 20, respectively, as values appropriate for our system. Please note that this step is complementary to the named entity recognition of OpenIE: Spotlight provides us with links to DBpedia, whereas OpenIE provides us with high-level entity categories that we use for text analysis.

Location Extraction. We next try to identify the location of the action in text triples by observing that words introducing a *grammatical case* in a sentence that are connected to a location often describe the verb's spatial location. Thus, we first extend each text triple into a *text quad* by specifying the location as unknown. Next, for each text quad where the object is a location (as indicated by entity recognition), we check whether the parse tree contains a grammatical case dependency between a word occurring in the quad's predicate and a word occurring in its object; if so, we move the quad's object to its location. For example, the object of ('Obama', 'met Trump in', 'White House') in Figure 2 has been classified as a location, and the parse tree contains a grammatical case dependency between the word 'House' occurring in the object and the preposition 'in' occurring in the predicate, and so we treat 'White House' as a location instead of an object. Please note that a location in the subject often does not specify the location of an action; for example, the subject of the sentence 'Oxford is a city' is a location, but 'Oxford' should not be used as a location in a quad since it does not describe the location of an action. We found no clear dependency pattern that could distinguish such cases and reliably extract location from subjects.

Passive Voice Correction. Passive voice can be problematical; for example, in ('Hawija', 'was bombed by', 'ISIS') from Figure 2, 'Hawija' is the subject and 'ISIS' is the object, which does not correctly reflect the intended meaning of the tweet. To correct such situations, for each text quad, we check whether the predicate contains a word that was classified by the POS tagger as a verb and that has (i) an outgoing *passive auxiliary modifier* dependency (to any other

[8] We abbreviate the actual resource `dbr:Islamic_State_of_Iraq_and_the_Levant`.

word), (ii) a *passive subject* dependency to a word occurring in the subject, and (iii) an *agent* dependency to a word occurring in the object; if so, we swap the subject and the object. In our example, 'was' is an auxiliary modifier, 'ISIS' is an agent, and 'Hawija' is a passive subject', so we apply the correction.

Entity Resolution. We next match the subject, object, and location of each text quad to the annotations of Spotlight. In case of an exact match we replace the component with the DBpedia resource, and otherwise we replace it with $\times$.

Verb Resolution. Since Spotlight does not handle verbs, we developed our own approach to verb resolution. First, we identify all verb occurrences in a tweet using POS tags. Next, we lemmatise each verb occurrence—that is, we substitute it with the verb's infinitive form (e.g., 'met' becomes 'to meet', 'bombed' becomes 'to bomb', and so on)—and then we search the tweet's parse tree for any phrasal verb particles connected to the verb's occurrence. Such a dependency indicates that the verb and the particle form an idiomatic phrase (e.g., 'take off' or 'sort out') and should be analysed together, so, whenever we find one, we concatenate the verb with the phrasal verb particle. We finally match the (possibly extended) verb occurrence to a WordNet synset; if several candidate synsets exist, we select the one that is most frequent according to the WordNet's statistics. The output of this part of our system is thus similar to that of Spotlight.

Finally, we resolve the predicates in the quads to the matched verbs. Unlike entities, which we resolved using exact matches, we substitute the predicate of a quad with a matched verb if the former *contains* the latter. This allows us to match 'was bombed by' in Figure 2 to the synset for 'to bomb'. Again, we replace predicates that could not be resolved with $\times$.

Quad Output. For each tweet, we return all quads except those containing only $\times$ markers. In addition, for each verb that was resolved to the tweet's text but could not be associated with a quad, we also return a fresh quad where the subject, object, and location are empty. Finally, we return the set of all entities that were detected by Spotlight but could not be matched to a quad.

5 Semantic Analysis

The Semantic Analysis component of `ArmaTweet` integrates DBpedia, WordNet, and the quads in a knowledge graph, and it evaluates complex event descriptions provided by the users. We next discuss the structure of the knowledge graph and the event descriptions, and we describe how we identify the tweet time series.

5.1 The RDF Knowledge Graph for Event Detection

We use RDF as the data model for the knowledge graph. Thus, the RDF versions of DBpedia and WordNet can be imported directly, and we encode tweet information using a simple schema. Each tweet is identified by a URI obtained from the tweet's ID; it is an instance of the `aso:Tweet` class; and data properties `aso:createdAt` and `aso:tweetText` specify the time of the tweet's creation and

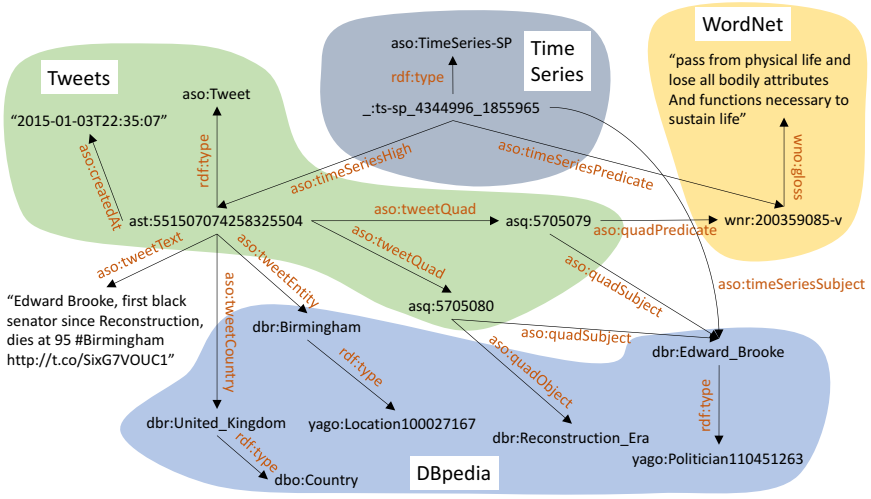

Fig. 3. A Fragment of the RDF Knowledge Graph

its text, respectively. A tweet can be associated with zero or more quads, each with at most one `aso:quadSubject`, `aso:quadPredicate`, `aso:quadObject`, and `aso:quadLocation` property value. Finally, a tweet can be associated with zero or more entities whose role in a sentence could not be determined (see Section 4).

Figure 3 shows the tweet `ast:551507074258325504` with two quads: one connects `dbr:Edward_Brooke` from DBpedia with the WordNet synset 'to die', and another connects `dbr:Edward_Brooke` with `dbr:Reconstruction_Era` (due to the imprecision of NLP analysis). Finally, the tweet is also directly associated with `dbr:Birmingham`, whose role in the sentence could not be determined.

The time series detected by Semantic Analysis component, each consisting of a summary and a set of tweets, are also stored in the knowledge graph. For example, `_:ts-sp_4344996_1855965` in Figure 3 is a tweet time series containing all tweets about Edward Brooke dying, and the Event Detection component (cf. Section 6) will extract from it zero or more events. Our system currently does not take into account that a person can die only once, and so it can potentially report multiple 'Edward Brooke dies' events. Each time series is classified according to the type of the summary information; currently, this includes subject–predicate (SP), predicate–object (PO), subject–country (SC), predicate–country (PC), and subject–predicate–country (SPC) time series. For example, the time series in Figure 3 is determined by a subject and a verb, and so it belongs to the `aso:TimeSeries-SP` class and the values of `aso:timeSeriesSubject` and `aso:timeSeriesPredicate` determine the time series summary.

5.2 Resolving Location in the Knowledge Graph

We observed that the granularity of the event location often varies between tweets; for example, tweets about the Charlie Hebdo attacks refer both to France

and Paris. To simplify event detection, we decided to aggregate event information at the country level. Thus, we extend the knowledge graph by resolving references to locations mentioned in tweets to the corresponding countries. For example, the tweet shown in Figure 3 refers to `dbr:Birmingham` so, since DBpedia contains the information that Birmingham is a city in the UK, we associate the tweet with `dbr:United_Kingdom` using the `aso:tweetCountry` property. Entities in tweet quads are resolved to countries in a similar vein.

5.3 Describing Complex Events and Extracting Time Series

Events of interest are described using conjunctive SPARQL queries that select the relevant quads. For example, queries (1) and (2) describe the 'politician dying' and the 'unrest in a country' events, respectively, where `aso:UnrestVerb` contains all verbs from WordNet that we identified as indicating unrest. The answer variables of each query determine the time series summary.

```
SELECT ?S wnr:200359085-v WHERE { ?Q aso:quadPredicate wnr:200359085-v .
                                  ?Q aso:quadSubject ?S . ?S rdf:type yago:Politician110451263 }   (1)

SELECT ?P ?C { ?Q aso:quadCountry ?C .
                                  ?Q aso:quadPredicate ?P . ?P rdf:type aso:UnrestVerb }   (2)
```

We next explain why querying quads is important. In particular, tweets often mention a politician and the verb 'to die', but not in a desired semantic relationship. For example, tweet `ast:551766588421312512` (not shown in Figure 3) says '@BarackObama @pmharper I'm just trying to get some realization, is school supposed to cause you so much stress&anxiety that you want to die?' and it is annotated with `dbr:Barack_Obama` and `wnr:200359085-v`, but, as one might expect from the text, there is no quad connecting the two resources. The lack of a semantic relationship, however, does not always indicate that a tweet is irrelevant to the event query. For example, tweet `ast:555598764589977600` (not shown in Figure 3) says 'Edward Brooke, first black US senator elected by popular vote, dies - Reuters', and it is annotated with `dbr:Edward_Brooke` and `wnr:200359085-v`, but, due to the complex sentence structure, the NLP component could not identify the semantic relationship correctly. In fact, our knowledge graph contains 44 tweets with quads matching 'Edward Brooke dies', as well as 111 additional tweets without the semantic relationship.

To exploit the knowledge graph as much as possible but without losing precision, our system proceeds as follows. It creates a tweet time series for each distinct result of a quad query (or, equivalently, for each distinct time series summary), to which it adds as 'high confidence' members all tweets containing a matching quad. Next, for each time series created in this way, the system adds to the time series as 'low confidence' members all tweets mentioning the relevant entities/predicates without the semantic relationship. For example, our system creates a time series for each distinct value of `?S` produced by query (1), and this includes `_:ts-sp_4344996_1855965` from Figure 3 that contains tweets `ast:551507074258325504` and `ast:555598764589977600` as 'high' and 'low confidence' members, respectively. In contrast, no time series is created for

dbr:Barack_Obama since our knowledge graph does not contain a quad matching query (1) where ?S is dbr:Barack_Obama. Intuitively, the presence of 'high confidence' tweets raises the importance of the 'low confidence' tweets, which helps compensate for the imprecision of the NLP analysis.

The Semantic Analysis component was realised using the RDFox system, which supports reasoning over RDF datasets using datalog rules. For each time series query, the user must provide the time series name and classify the query according to the summary type, and then the query is converted into a datalog rule that creates the tweet time series and identifies the 'high confidence' tweets. For example, query (1) is named aso:PoliticianDying and classified as a subject–predicate query, and so it is converted into the following datalog rule:

```
[?TS, rdf:type, aso:PoliticianDying], [?TS, aso:timeSeriesSubject, ?S],
[?TS, aso:timeSeriesVerb, wnr:200359085-v], [?TS, aso:timeSeriesHigh, ?TW] :-
    [?TW, aso:tweetQuad, ?Q], [?Q, aso:quadSubject, ?S],
    [?S, rdf:type, yago:Politician110451263], [?Q, aso:quadPredicate, wnr:200359085-v],
    BIND(SKOLEM("ts-sp", ?S, wnr:200359085-v) AS ?TS) .
```
(3)

This rule uses the datalog syntax of RDFox, which supports calling SPARQL builtin functions in its body. The SKOLEM function is an RDFox-specific extension that creates a blank node uniquely determined by the function's parameters, thus simulating function symbols from logic programming. Thus, for each value of ?S, rule (3) assigns to ?TS a unique blank node that identifies the time series, and its head atoms then attach to ?TS the relevant information and the 'high confidence' tweets. A fixed (i.e., independent from the queries) set of rules then identifies the 'low confidence' members of each time series by selecting tweets that mention all entities/predicates from the time series summary, but without the semantic relationship. For example, for subject–predicate time series, these rules select all tweets that mention the subject and the predicate outside a quad.

6 Event Detection

The Event Detection component accepts as input the tweet time series produced by the Semantic Analysis component, and it identifies zero or more associated events. This is done using the Seasonal Hybrid ESD (S-H-ESD) test [28] developed specifically for detecting anomalies in Twitter data. The algorithm is given a real number p between 0 and 1, a set of time points T, and a real-valued function $x : T \to \mathbb{R}$ that can be seen as a sequence of observations of some value on T where $x(t)$ is the value observed at time $t \in T$. The algorithm identifies a subset T_a of T of time points at which the value of x is considered to be anomalous, while ensuring that $|T_a| \leq p \cdot |T|$ holds; thus, p is the maximal proportion of the time points that can be deemed anomalous. Roughly speaking, the S-H-ESD test first determines the periodicity/seasonality of the input data; next, it splits the data into disjoint windows each containing at least two weeks of data; finally, for each window, it subtracts from x the seasonal and the median component and applies to the result the Extreme Student Derivative (ESD) test—a well-known anomaly detection technique. Twitter is currently using this technique

Table 1. Evaluation Results by Event Category

Event Category	Type	Total Events	Positive Instances by Relevance R3	R3+R2	R3–R1
Aviation accident	SP	84	44 (52%)	51 (61%)	64 (76%)
Cyber attack on a company	PO	129	20 (16%)	42 (33%)	57 (44%)
Capital punishment in a country	PC	153	47 (31%)	67 (44%)	92 (60%)
Militia terror act	SP	220	92 (42%)	125 (57%)	141 (64%)
Politician dying	SP	111	76 (68%)	80 (72%)	85 (77%)
Politician visits a country	SPC	44	29 (66%)	36 (82%)	44 (100%)
Unrest in a country	PC	200	125 (63%)	133 (67%)	148 (74%)
Total:		941	433 (46%)	534 (57%)	631 (67%)

on a daily basis to analyse their server load. `ArmaTweet` uses the open-source implementation of this test from the R statistical platform.[9]

To apply the S-H-ESD test, each tweet time series is converted into a sequence of temporal observations by aggregating the tweets by day—that is, the set T corresponds to the set of all days with at least one tweet, and the value of $x(t)$ is the number of (both 'high' and 'low confidence') tweets occurring on the day $t \in T$. We then run the S-H-ESD test with $p = 0.05$—that is, at most 5% of the time points can be deemed anomalous. Moreover, we configured the algorithm to detect only positive anomalies (i.e., cases where the number of tweets is above the expected value), which is natural for event detection.

7 Evaluation

Evaluating `ArmaTweet` was difficult since there is no ground truth: a list of all relevant events does not exist, so we could not determine the recall of our technique. Thus, we focused on determining the precision and the benefits of semantic event detection. We next present our experimental setup and discuss our findings.

We processed 195.7 M tweets in English collected in the first half of 2015 using Twitter's streaming API (which returns about 1% of all tweets). The NLP component extracted 14.5 M quads from 12.8 M tweets (i.e., 6% of the input). Most quads have two components: 6.2 M quads contain a predicate and an object, and 5 M quads contain a subject and a predicate; the remaining 0.7 M quads have three components, and no quads have four components. About 0.5 M quads contain location information. Integrating the quad information with DBpedia and WordNet produced a knowledge graph containing a total of 725.8 M triples, which increased to 800 M triples after applying the semantic analysis rules.

Determining Complex Events. We consulted the Wikipedia page for 2015[10] to identify interesting concrete events, which provided us with a starting point

[9] http://github.com/twitter/AnomalyDetection
[10] http://en.wikipedia.org/wiki/2015

for a series of workshops in which we identified events and event types of interest to armasuisse customers. We eventually settled on the seven complex event categories shown in Table 1. We made sure that our categories cover many different types of event summary (i.e., subject–verb, verb–object, etc.).

Creating Category Queries. For each event category, we constructed a semantic query as follows. We first identified the entities from our example events on Wikipedia (e.g., `dbr:Edward_Brooke`), which we then looked up in DBpedia to identify their types (e.g., `yago:Politician110451263`). Next, we queried our knowledge graph for the verbs occurring together with the sample entities in the tweets. We ranked these verbs by the frequency of their occurrence, and then selected those best matching the event category. Finally, we formulated the category query and tested it on example events. Most queries capture the meaning of the categories directly, apart from the 'Aviation incident' query where, to select useful data, we ask for a subject of type 'airline' and a verb indicating a crash. Creating the queries took about four person-days of an expert in semantic technologies, and optimising this process is the main topic for our future work.

Event Validation. By evaluating the event categories over the knowledge graph and detective events as discussed Sections 5 and 6, we identified a total of 941 events (see Table 1), which we validated manually—that is, we determined whether the reported event is a positive instance. This, however, turned out to be surprisingly challenging. First, we could often not verify whether the event really happened, so we decided to just evaluate whether the retrieved tweets correctly talk about the event; we justify this choice by noting that detecting 'invented' events could also be very important to security analysts. Second, some events happened in the past (e.g., the anniversary of Robert Kennedy's assassination was widely discussed on Twitter), but we decided to count these as positive instances as well since they are also likely to be of interest. Third, in some cases the retrieved events were only partially relevant to the query, and so we assigned each event one of the following three relevance scores:

- R3 are clear positive instances of the category in question;
- R2 are positive instances where the entity resolution (e.g., `dbr:British_Raj` vs. `dbr:India`) or the subject–object relationships (e.g., 'ISIS attacked X' vs. 'X attacked ISIS') in the event summary are incorrect;
- R1 are events with a 'fuzzy' relationship to the category (e.g., 'ISIS kills X' or 'policeman killed' for the 'Unrest in a country' category); and
- R0 are events with no relevance to the event category.

Results. Table 1 shows the total number of detected events per category and the numbers of positive instances for different relevance scores. As one can see, precision varies considerably across categories. Visits and deaths of politicians could be reliably detected: our NLP component seems very effective on the relevant tweets, and type filtering seems very effective at identifying the appropriate entities. In contrast, detecting cyber attacks is difficult: our query searches for 'company hacked', but the verb 'to hack' often means 'to cut' or 'to manage' so the query retrieved many irrelevant tweets (e.g., about a blogger being stabbed).

A particular problem for `ArmaTweet` was to correctly differentiate the subject from the object of an action: the approach to passive voice detection we described in Section 4 was effective, but should be further improved. Moreover, precision often suffered due acronyms; for example, 'APIs' (i.e., 'Application Programming Interfaces') was resolved to 'Associated Press'. Finally, popular entities posed a particular problem. For example, ISIS appears in a great number of tweets, which increases the likelihood of incorrect event recognition; in contrast, Boko Haram is not that well known and thus seems to be mainly mentioned in tweets reporting terrorist activity. We plan to further investigate ways to 'normalise' the tweet time series based on the 'popularity' of the entities involved.

8 Conclusion

We have presented `ArmaTweet`—a system developed by armasuisse and the Universities of Fribourg and Oxford for semantic event detection on Twitter. The system represents the tweets' contents in an RDF knowledge graph, thus allowing users to precisely describe the events of interest. The results of our evaluation show that `ArmaTweet` can detect events such as 'politician dying' and 'militia terror act', which cannot be detected by conventional keyword-based methods. We see two main challenges for future work. First, to help users describe complex events, we will develop adequate user interfaces, as well as investigate ways to extract semantic queries from example tweets. Second, we plan to improve the precision of the NLP component, particularly focusing on the correction of passive voice and the quality of entity resolution in the presence of acronyms.

References

1. Angeli, G., Premkumar, M., Manning, C.: Leveraging Linguistic Structure For Open Domain Information Extraction. In: ACL. pp. 344–354 (2015)
2. Becker, H., Chen, F., Iter, D., Naaman, M., Gravano, L.: Automatic Identification and Presentation of Twitter Content for Planned Events. In: ICWSM. pp. 655–656 (2011)
3. Becker, H., Naaman, M., Gravano, L.: Beyond Trending Topics: Real-World Event Identification on Twitter. In: ICWSM. pp. 438–441 (2011)
4. Benson, E., Haghighi, A., Barzilay, R.: Event Discovery in Social Media Feeds. In: HLTCon 2011. pp. 389–398 (2011)
5. Bontcheva, K., Derczynski, L., Funk, A., Greenwood, M.A., Maynard, D., Aswani, N.: TwitIE: An Open-Source Information Extraction Pipeline for Microblog Text. In: RANLP. pp. 83–90 (2013)
6. Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving Efficiency and Accuracy in Multilingual Entity Extraction. In: I-SEMANTICS. pp. 121–124 (2013)
7. Farzindar, A., Khreich, W.: A Survey of Techniques for Event Detection in Twitter. Computational Intelligence 31(1), 132–164 (2015)
8. Gu, H., Xie, X., Lv, Q., Ruan, Y., Shang, L.: ETree: Effective and Efficient Event Modeling for Real-Time Online Social Media Networks. In: WI. pp. 300–307 (2011)
9. Kong, L., Schneider, N., Swayamdipta, S., Bhatia, A., Dyer, C., Smith, N.: A Dependency Parser for Tweets. In: EMNLP. pp. 1001–1012 (2014)

10. Lee, R., Sumiya, K.: Measuring Geographical Regularities of Crowd Behaviors for Twitter-based Geo-social Event Detection. In: LBSN. pp. 1–10 (2010)
11. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P., ..., Bizer, C.: DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web 6(2), 167–195 (2015)
12. Li, R., Lei, K.H., Khadiwala, R., Chang, K.C.C.: TEDAS: A Twitter-based Event Detection and Analysis System. In: ICDE. pp. 1273–1276 (2012)
13. Long, R., Wang, H., Chen, Y., Jin, O., Yu, Y.: Towards Effective Event Detection, Tracking and Summarization on Microblog Data. In: WAIM. pp. 652–663 (2011)
14. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The Stanford CoreNLP Natural Language Processing Toolkit. In: ACL. pp. 55–60 (2014)
15. Massoudi, K., Tsagkias, M., de Rijke, M., Weerkamp, W.: Incorporating Query Expansion and Quality Indicators in Searching Microblog Posts. In: Prof. ECIR. pp. 362–367 (2011)
16. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM 38(11), 39–41 (1995)
17. Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., Smith, N.: Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. In: HLTCon. pp. 380–390 (2013)
18. Petrović, S., Osborne, M., Lavrenko, V.: Streaming First Story Detection with application to Twitter. In: HLTCon. pp. 181–189 (2010)
19. Phuvipadawat, S., Murata, T.: Breaking News Detection and Tracking in Twitter. In: WI-IAT. pp. 120–123 (2010)
20. Popescu, A.M., Pennacchiotti, M.: Detecting Controversial Events from Twitter. In: CIKM. pp. 1873–1876 (2010)
21. Ramakrishnan, N., Butler, P., Muthiah, S., Self, N., Khandpur, R., Saraf, P., ..., Mares, D.: 'Beating the news' with EMBERS: forecasting civil unrest using open source indicators. In: KDD. pp. 1799–1808 (2014)
22. Ritter, A., Clark, S., Mausam, Etzioni, O.: Named Entity Recognition in Tweets: An Experimental Study. In: EMNLP. pp. 1524–1534 (2011)
23. Ritter, A., Mausam, Etzioni, O., Clark, S.: Open Domain Event Extraction from Twitter. In: KDD. pp. 1104–1112 (2012)
24. Rospocher, M., van Erp, M., Vossen, P., Fokkens, A., Aldabe, I., Rigau, G., ..., Bogaard, T.: Building event-centric knowledge graphs from news. Journal of Web Semantics 37, 132–151 (2016)
25. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In: WWW. pp. 851–860 (2010)
26. Sankaranarayanan, J., Samet, H., Teitler, B.E., Lieberman, M., Sperling, J.: TwitterStand: News in Tweets. In: ACM-GIS. pp. 42–51 (2009)
27. Signorini, A., Segre, A., Polgreen, P.: The Use of Twitter to Track Levels of Disease Activity and Public Concern in the U.S. during the Influenza A H1N1 Pandemic. PLoS ONE 6(5), 1–10 (2011)
28. Vallis, O., Hochenbaum, J., Kejariwal, A.: A Novel Technique for Long-Term Anomaly Detection in the Cloud. In: HotCloud. pp. 1–6 (2014)
29. Weng, J., Lee, B.S.: Event Detection in Twitter. In: ICWSM. pp. 401–408 (2011)
30. Zhao, L., Chen, F., Dai, J., Hua, T., Lu, C.T., Ramakrishnan, N.: Unsupervised Spatial Event Detection in Targeted Domains with Applications to Civil Unrest Modeling. PLoS ONE 9(10), 1–12 (2014)
31. Zhou, X., Chen, L.: Event detection over twitter social media streams. VLDB Journal 23(3), 381–400 (2014)