Pattern Discovery and Anomaly Detection via Knowledge Graph

Bin Jia, Cailing Dong, Z. Chen
Intelligent Fusion Technology, Inc
Germantown, MD USA
{bin.jia,cailing.dong, zhijiang.chen}@intfusiontech.com

Kuo-Chu Chang
Systems Engineering and Operations Research
George Mason University
Fairfax, VA USA
kchang@gmu.edu

Nichole Sullivan, Genshe Chen
Intelligent Fusion Technology, Inc
Germantown, MD USA
{nichole.sullivan, gchen}@intfusiontech.com

Abstract—**In this paper, we developed a pattern discovery and anomaly detection system using a knowledge graph constructed by integrating data from heterogeneous sources. Specifically, the knowledge graph is constructed based on data extracted from structured and unstructured sources. Besides the extracted entities and relations, the knowledge graph finds hidden relations via link prediction algorithms. Based on the constructed knowledge graph, the normalcy model for entity, action, and triplets are established. The information of the incoming streaming data is extracted and compared to the normalcy model in order to detect abnormal behaviors. In addition, we apply the lambda framework to enable a computationally scalable algorithm for pattern discovery and anomaly detection in a big data environment. Real time tweets data are used for evaluation and preliminary results show promising performance in detecting abnormal pattern and activities.**

Keywords—Knowledge Graph, Knowledge Representation, Multi-INT fusion, Pattern Discovery, Anomaly Detection

I. Introduction

Pattern discovery and anomaly detection are critical elements in Multi-INT fusion for surveillance and situation assessment. To achieve situational awareness, various sensors or data sources are available, including conventional sensors such as a video camera, radar, and infrared sensor, as well as non-traditional open sources, such as tweets. The conventional sensors often provide structured measurements to users. Machine learning or data mining techniques are then applied to extract frequent patterns. The learning algorithms use structured data to learn the mapping from sensor input to output patterns. Based on the learned models, various pattern discovery algorithms are utilized to detect the abnormal behaviors. For unstructured data received from open sources, such as text, it is often difficult to apply the traditional learning algorithm directly to extract patterns. In addition, to fuse information from multiple sources, a unified representation of the information for both unstructured data and structured data is required. To achieve this purpose, we propose to use knowledge graph (KG) model.

Knowledge graph, also called graph structured knowledge bases (KBs), stores facts by describing entities and their relations with a unified representation of knowledge from various resources. It also facilitates advanced learning algorithms to predict potential links between identifiers and hidden attributes of entities. KG is therefore ideally suitable for discovering unusual patterns and detecting abnormal activities. In recent years, many large knowledge graph databases were developed, such as Freebase [1], YAGO [2], NELL [3], DBpedia [4], WordNet [4] and Google Knowledge Graph [6]. Due to the increasing data volume, the implementation of the scalable algorithm is highly desirable.

The main contribution of this paper is to develop methodologies and implement a KG reasoning system to discover abnormal patterns and unusual activities from unstructured data received from open sources. While the system was developed and demonstrated for a specific scenario, the methodologies and the system framework are general and can be easily extended and applied to different domains. The paper is organized as follows. Section II introduces the knowledge graph representation. The knowledge graph construction, mining, and inference are described in Section III, followed by some examples given in Section IV. The detailed system implementation and case study are presented in Section V. Section VI discusses some concluding remarks.

II. Knowledge Representation

Knowledge graph models information using entities and their relations. In this paper, we use subject-predicate-object (SPO) triples following the standard of World Wide Web Consortium (W3C) Resource Description Framework (RDF) [7] to represent the existence of facts. The subject and object are named entities while the predicate describes the relations between the entities. Depending on the relationships, many triples may be connected to each other. Hence, a sizable graph can be constructed from a large number of triples extracted from unstructured texts. Note that different kinds of attributes of the entities and different relations between entities can be defined for the knowledge graph. In the following, we describe the basic steps of building a knowledge graph.

A. Entity Extraction

As an essential step of information extraction, Named-Entity Recognition (NER) aims to locate and classify entities in document into pre-defined categories such as locations, things, persons, and the names of organizations. NER systems are developed using statistical models or linguistic grammar-based techniques. Typically, statistical NER systems require a large amount of training data annotated manually, which is labor-intensive. Semi-supervised approaches have been proposed to reduce part of the heavy annotation effort, however, they usually produce less accurate results. There are three standard approaches to NER. The first type of approach is based on manually designed regular expression. Obviously, this approach lacks generalizability and scalability. The second approach is to build classifiers on a large training dataset, such as the generative classifier with Naïve Bayes and the discriminative classifier based on Maximum Entropy Models. The third approach relies on building context sequence models. Clearly, to accurately produce the NER label for each entity, it would beneficial to consider the contexts of each word in a sequence. Some of the most popular context sequence models include Hidden Markov Model (HMM) [8], Conditional Markov Model (CMM) [9], and Conditional Random Fields (CRFs) [10]. Instead of predicting a single label for a single data sample, sequential modeling approaches take context into consideration and predict sequences of labels from sequences of input samples. The resulting collections of named entities provide essential elements in a given scenario such as people, locations, and organizations extracted from a massive text source.

B. Relation Extraction

In addition to the collection of entities, another critical step is to explore and identify the relations between them. The task of Relation Extraction (RE) is to detect and classify the semantic relationships between named entities within a set of text/documents. The approaches of extracting relations are similar to NER approaches, i.e., either using a regular expression to associate entities with relations, or building classifiers based on training data. Primarily, RE explores typical relations between two entities, such as relations between two people or relations between a person and an organization. However, there are other non-traditional types of relations between entities or non-entities, which could be critical to anomaly detection. This is the task of Open Information Extraction (Open IE) [11]. Open IE aims to extract relation propositions from the plain text without using pre-defined ontology or schema. The established relation, typical or otherwise, is the link between two arguments. In this paper, we are interested in the discovery of subject-predicate-object (SPO) triples from free texts for relation extraction, identification, and confirmation, where subject and object are not restricted to be entities discovered by NER and the predicate represents any relation between them.

C. Relation Elimination and Entity Resolution

OpenIE tends to produce many redundant relations (SPO triples) and identity noises for the same set of entities. It is necessary to eliminate redundant triples and resolve identity uncertainty to develop a scalable and credible knowledge graph. We designed a relation reduction algorithm to eliminate the redundant candidate SPO triples. This algorithm mainly focuses on the part-of-speech (POS) tags of the words in the triples. POS-tag based rules are defined to classify composite sentences from single sentences. The most concise simple SPO triple is selected to represent the relation between subject and object in a given sentence. In addition, we replace the original word in the selected triple with its corresponding lemmatized form to reduce the noises caused by verb tenses or singular vs. plural nouns.

Entity resolution is the process of identifying and grouping different manifestations of the same real-world objects from various sources. It aims to link and group different entity mentions for the same underlying entity. For instance, in our Ukraine dataset to be described later, “Ukraine pres Poroshenko”, “president Poroshenko” and “pres @poroshenko” all refer to the same entity “Petro Poroshenko”. It is very important to reduce the noises caused by this type of identity duplications for constructing a proper knowledge graph. We employed large knowledge bases (KBs), such as Google Knowledge Graph, to resolve this issue. Specifically, for each subject and object in the extracted SPO triples, we locate and apply the entity name and corresponding entity identifier in Google Knowledge graph or Wikipedia. Using this approach, we largely reduced the noises caused by multi-representation of the same word or entity, thus the final SPO triples are more representative to build the knowledge graph.

III. Knowledge Graph: Construction, Mining and Inference

A. Knowledge Graph Construction

Knowledge graphs (KGs) can be considered as KBs that model and store information in the form of entities and their relations using a graph structure [7]. There are typically two assumptions regarding KGs. With a Closed World Assumption (CWA), any relation that is not represented by existing triples is considered false. However, a non-existing triple is interpreted as unknown under the Open World Assumption (OWA). Seemingly, OWA makes more sense as KGs are known to be very sparse and incomplete. KBs are typically constructed using the four different methods described in Table I [7].

In this paper, we develop a domain-specific knowledge graph with facts represented by standard SPO triples extracted from Twitter messages. We apply relation elimination and entity resolution to manage the knowledge base. Subjects and objects are entities and represented as nodes in the KGs. The relationships between entities, i.e., the relations linking subjects to objects, are represented as edges. Figure 1 shows an example of a partial knowledge graph for a specific domain considered in this paper.

Table I. Knowledge Graph Construction

Methods	**Triple Creation Methods**	**Typical Examples**
Curated	Manually developed by a group of experts	WordNet, UMLS
Collaborative	Manually generated by a group of collaborative volunteers	Wikidata, Freebase
Automated Semi-Structured	Automatically extracted from semi-structured text via hand-crafted or learned rules	YAGO, DBPedia, Freebase
Automated Unstructured	Automatically extracted from unstructured text using natural language processing or machine learning techniques	NELL, DeepDive

B. Knowledge Graph Learning and Prediction

By nature, a knowledge graph is sparse, highly incomplete, and noisy. To overcome these issues, "statistical relational learning" (SRL) is applied to predict missing links and identify relations between nodes. To describe statistical learning model of a knowledge graph, we define the following notations [7]. We denote $E = \{e_1, \cdots, e_{N_e}\}$ as the set of all entities and $R = \{r_1, \cdots, r_{N_r}\}$ as the set of all relation types in the knowledge graph. Possible triples can be defined by $x_{ijk} = (e_i, r_k, e_j)$. The binary random variable $y_{ijk} \in \{0,1\}$, where

$$y_{ijk} = \begin{cases} 1 & \text{if the triple } (e_i, r_k, e_j) \text{ exists} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The goal is to predict new facts (unknown or hidden relations) given existing facts in a KG and to reason about the noisy "facts" extracted from various sources. The link prediction problem can be formulated using a probabilistic approach where each triple is assigned a probability of being true. The model predicts the existence of x_{ijk} via a score function, $f(x_{ijk}; \theta)$. The score function represents the confidence of the existence of the triple x_{ijk} given the unknown parameters [12]. More specifically, the link prediction problem can be formulated as an optimization problem by maximizing the log-likelihood function,

$$\max_{\theta} \sum_{n=1}^{N_d} \log B\left(y^n \mid \sigma\left(f\left(x^n; \theta\right)\right)\right) \tag{2}$$

where N_d is the number of observed triples, $\sigma(\cdot)$ is the logistic function and B is the Bernoulli distribution given by

$$B(y \mid p) = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases} \tag{3}$$

θ is the unknown parameters to be learned in the training process, and $f(\cdot)$ is the score function which can be defined based on different criterion. One popular approach called *RESCAL* is based on the bilinear model [12],

$$f_{ijk}^{RESCAL} = \mathbf{e}_i^T \mathbf{W}_k \mathbf{e}_j \tag{4}$$

where $\mathbf{e}_j$ and $\mathbf{e}_j$ are the latent feature representation of the entity e_i and e_j, respectively. $\mathbf{W}_k$ represents the bilinear weight matrix of relation r_k. Note that Eq. (4) is based on the assumption that y_{ijk} are conditionally independent given latent features associated with the entities.

It has been shown the *RESCAL* method is scalable for a large knowledge graph. Thus, in this paper, we employ *RESCAL* model to predict potential relations between entities.

C. Knowledge Graph Mining

Knowledge graph mining is to identify frequent items and determine associations between items in the data. Based on the extracted and inferred SPO triples, we apply the Apriori algorithm [13] to identify frequent subject(S), object(O), predicate (P) and any combination of them. Specifically, Apriori is an influential algorithm for learning association rules and mining frequent item sets. It starts with identifying the frequent individual items in the given data set. The items are extended to larger sets if they appear sufficiently often. The identified frequent items are then used to determine association rules, that is, the frequent associations between items.

Denote $I = \{i_1, i_2, \ldots i_n\}$ as a set of items in a transactions T, the threshold of support value σ determines that the item or item set occurred in T, at least σ times, are considered to be frequent. Apriori uses a hash tree structure and applies breadth-first search to count candidate item sets efficiently. The pseudo code for Apriori algorithm is summarized below [13]. The input to the algorithm is the data set T and the threshold σ. At the beginning, all the single items with support value >= σ are qualified as output. The algorithm iteratively generates the candidate sets at each step, from the large item sets obtained from the preceding level, to find frequent item sets and use them to generate association rules based on the minimum confidence threshold.

```
Algorithm 1: Apriori Algorithm
Apriori (T, σ)
$L_1 \leftarrow$ {large 1-itemsets}
$k \leftarrow 2$
while $L_k \neq \emptyset$:
    $C_k \leftarrow \{a \cup \{b\} \mid a \in L_{k-1} \wedge b \neq a\} - \{c \mid \{s \mid s \subseteq c \wedge s| = k-1\} \notin L_{k-1}\}$
    for transactions t ∈ T:
        $C_t \leftarrow \{c \mid c \in C_k \wedge c \subseteq t\}$
        for candidates $c \in C_t$:
            $count[c] \leftarrow count[c] + 1$
    $L_k \leftarrow \{c \mid c \in C_k \wedge count[c] \geq \sigma$
    $k \leftarrow k + 1$
Return $\cup_k L_k$
```

Temporal information is another feature that distinguishes streaming data from traditional structured/unstructured data. To explore such information, we use PrefixSpan [13] algorithm to find frequent sequential patterns from streaming data. The detailed implementation of PrefixSpan is described below. The implementation details are based on the definitions of *subsequence/super-sequence*, *prefix* and *projection*.

- Given two sequences $\alpha = < \alpha_1, \alpha_2, \ldots \alpha_n >$ and $\beta = < \beta_1, \beta_2, \ldots \beta_m >$, α is called a **subsequence** of β if there exist integers $1 \leq j_1 < j_2 < \cdots < j_n \leq m$ such that $\alpha_1 \sqsubseteq \beta_{j_1}, \alpha_2 \sqsubseteq \beta_{j_2}, \ldots, \alpha_n \sqsubseteq \beta_{j_n}$. β is a **super-sequence** of α.
- Given two sequences $\alpha = < \alpha_1, \alpha_2, \ldots \alpha_n >$ and $\beta = < \beta_1, \beta_2, \ldots \beta_m >$ with $m \leq n$, sequence β is called a **prefix** of α if and only if: (1) $\beta_i = \alpha_i$ for $i \leq m - 1$; (2)

$\beta_m \subseteq \alpha_m$; (3) all the items in ($\alpha_m - \beta_m$) are alphabetically after those in β_m.

- Given two sequences α and β such that α is a subsequence of β, a subsequence β' of sequence β is called a **projection** of β w.r.t α prefix if and only if: (1) β' has prefix α, (2) there exist no proper super-sequences β'' of β' such that β'' is a subsequence of β and also has prefix α.

```
Algorithm 2: PrefixSpan Algorithm

Input: A sequence dataset S, and the minimum support threshold min_support
Output: The complete set of sequential patterns
Method: PrefixSpan (<>, 0, S)
Subroutine PrefixSpan ($\boldsymbol{\alpha}$, len, S|$\boldsymbol{\alpha}$):
1: Scan S|$\boldsymbol{\alpha}$ once and find the set of frequent items $\boldsymbol{\beta}$ such that:
      a. $\boldsymbol{\beta}$ can be assembled to the last element of $\boldsymbol{\alpha}$ to from a sequential pattern; or
      b. $<\boldsymbol{\beta}>$ can be appended to $\boldsymbol{\alpha}$ to form a sequential pattern.
2: For each frequent item $\boldsymbol{\beta}$, append it to $\boldsymbol{\alpha}$ to form a sequential pattern $\boldsymbol{\alpha}'$ and output $\boldsymbol{\alpha}'$;

For each $\boldsymbol{\alpha}'$ , construct $\boldsymbol{\alpha}'$ -projected dataset S|$\boldsymbol{\alpha}'$ , and call PrefixSpan ($\boldsymbol{\alpha}'$ , len+1, S|$\boldsymbol{\alpha}'$ )
```

IV. Examples

A. Knowledge Graph Construction

We collected a special Twitter data set about Ukraine from Live Universal Awareness Map ("Liveuamap") from January 1st, 2017 to September 30th, 2017. The original publisher of the tweets, the retweet information as well as the embedded entities (photos, videos, links) are also retrieved.

We employed CoreNLP toolkit [14] to extract named entities from our Ukraine dataset and used its Open IE package to extract all the SPO triples in each message in the data set. As the candidate SPO triples produced by Open IE have many redundancies, we applied our relation elimination and entity resolution methods to resolve SPO triples to construct the knowledge graph.

Table II shows an example of relation elimination on SPO triples. The top panel shows the original SPO triples produced by Open IE package for each message. The bottom panel shows the final SPO triple(s) after elimination process. As we mentioned earlier, entity resolution is an important step to build a knowledge graph. Table III shows an example of applying our entity resolution process on the data set. At least 8 different entity mentions exist in the data set for the same entity "Petro Poroshenko". Obviously, entity resolution helps to remove many noises and enable efficient graph knowledge construction and data analytics.

TABLE II. EXAMPLE OF RELATION ELIMINATION

Original SPO triples
• Poroshenko discussed military cooperation • Poroshenko discussed cooperation during Zapad2017 • Poroshenko discussed cooperation • Reform is overdue • Reform is long overdue • Comprehensive reform is long overdue • Comprehensive reform is overdue
Final SPO triples
• <Poroshenko, discuss, cooperation> • <reform, is, overdue>

TABLE III. EXAMPLE OF ENTITY RESOLUTION

Original Entities	Final Entities in **Google KG**
• petro Poroshenko • Ukraine pres Poroshenko • ukraine @poroshenko • poroshenko • president Poroshenko • pres Poroshenko	**Name:** Petro Poroshenko **Identifier:** kg:/m/08w60w **url:** https://en.wikipedia.org /wiki/Petro_Poroshenko

We constructed a small part of the knowledge graph built based on August messages of the Ukraine data set. While there are some isolated SPO triples not shown in the figure, most of the SPO triples are connected to one another. The hub nodes in the knowledge graph with many incoming links and outgoing links are "Russia", "Crimea", "Petro Poroshenko" and "Ukraine". Clearly, they represent the main topics about Ukraine crisis. Therefore, the activities extracted from the data sets are centered around these entities. Given "Petro Poroshenko" as an example, part of the activities about him are demonstrated in the sub-graph of the overall knowledge graph in Figure 1. We can see that the president of Ukraine, i.e., Petro Poroshenko, more often serves as the subject in the SPO triples, due to his special identity. In the future, we plan to employ WordNet [5] to cluster those links, to better represent and track these types of activities. In addition to the objects directly linked to "Petro Poroshenko", we are also able to obtain knowledge of different relations about those objects. From the examples, it is clear the knowledge graph can provide a global picture about the entity-of-interest, and provide existing "facts" for verification of potential anomalies.

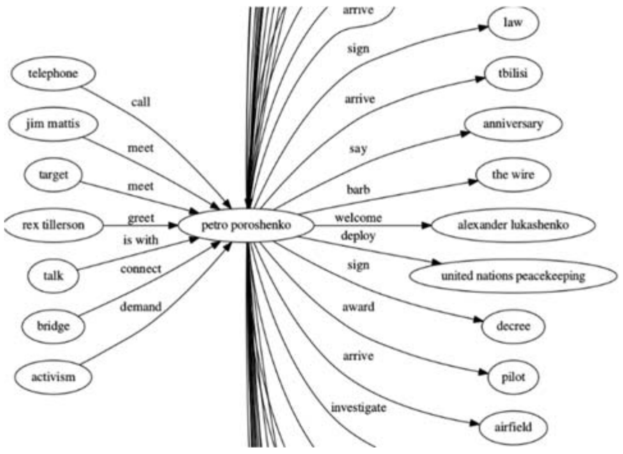


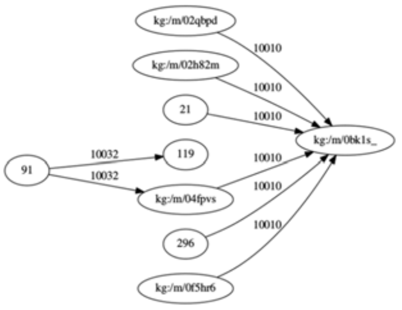
Fig. 1. Knowledge graph about "Petro Poroshenko"

B. Pattern Mining and Abnormal Detection

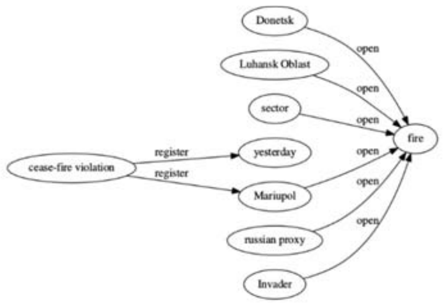
We collected all the SPO triples in a each time window andapplied Apriori algorithm on the collection with a minimum support value σ. By ranking the frequent size-1 items from high to low, we obtained the most important subjects, objects and predicates in the knowledge graph. The size-2 frequent items can reveal the associations of subject-object, subject-predict and predict-object, and the size-3 frequent items are the frequent SPO triples, which represent some important activities and events.

For example, Figure 2 shows the frequent SPOs regarding to activity "open fire" in August messages. All the nodes in Figure 3(a) are represented by their identifiers. We can see some of the node identifiers are numbers, while some of them are represented by string literals. This is the result of combining the local dynamic knowledge graph with the general Google knowledge graph (Google KG). Specifically, for those nodes (subjects or objects in SPO triples) which can be mapped to an entity in Google KG in the entity resolution process, we denote them using the corresponding Google KG identifier, i.e., in the form of "Kg://xxxxxx". While, for those subjects or objects that cannot map to Google KG, we represent them using unique identifiers in our local KG. By observing the frequent SPO triples in Figure 2(b), we can discover some patterns. For example, the most frequent locations having the activities "open fire". It is inevitable that some vague relations such as "sector open fire", "cease-fire violation register yesterday" are present in the KG. Further refining SPO extraction process is one of our future work. It will be also beneficial to consider temporal and spatial information to refine the knowledge graph in the future.

To explore the hidden sequential patterns, we collect all the messages regarding a specific entity or activity within each month and create a sequence using the monthly data. As "open fire" is a frequent activity in our dataset, and we are interested in knowing how different entities are connected regarding to this activity, we conduct frequent sequential mining using PrefixSpan algorithm on the sequential records. The results of frequent sequential pattern mining are shown in Table IV. The sequential pattern "Invader -> militant" indicates the activity "invader open fire" is usually followed by the activity "militant open fire". The sequential pattern "Donetsk -> Donetsk -> Donetsk" with a frequency value 4 indicates that Donetsk is the major battlefield, as the activity "Donetsk open fire" happened consecutively and frequently in time. To better understand the battle pattern, it can be explored by analyzing the sequential location patterns regarding to this activity. By converting the (latitude, longitude) pair into specific address in village or city level, we further obtain the knowledge that Zaitseve and Avdiivka are the major cities of Donetsk province in Ukraine with warfare. The warfare battle plan tended to be from Zaitseve to Avdiivka and then returned to Zaitseve. Apparently, ffrequent sequential patterns provide potentially important information to predict future activities.



(a) Nodes represented by identifiers



(b) Nodes represented by name

Fig. 2. Frequent patterns about activity "open fire"

TABLE IV. FREQUENT SEQUENTIAL PATTERNS ABOUT "OPEN FIRE"

Sequential Entities	Frequency
Invader -> militant	3
Donetsk -> Donetsk -> Donetsk	4
Donetsk -> Invader	4
48.417,38.033 -> 48.124,37.769 (Zaitseve, Donetsk → Avdiivka, Donetsk)	3
48.124,37.769 -> 48.124,37.769 (Avdiivka, Donetsk -> Avdiivka, Donetsk)	5
48.124,37.769 -> 48.417,38.033 (Avdiivka, Donetsk → Zaitseve, Donetsk)	3

After obtaining the frequent item sets on the constructed KG, the system can automatically track the activities of the most important entities (subjects/objects) and their relations (predicates). The activities are tracked continuously with corresponding locations. In the meantime, potential anomalies can be discovered and regarded as subjects or objects associated with unusual predicates, unusual subjects/objects associated with a given predicate, or uncommon locations and sudden location change.

As Ukraine capital "Kiev (Kyiv)" is a frequent entity in the extracted frequent item sets, we track all the activities about it. In Table V, we list the activities associated to Kiev from late August to early September in time order. As Kiev itself is a location, the detailed geoinformation is omitted here. Regarding this specific location entity, potential anomalies are "sensitive" subjects such as "wreckage" and "car bomb", or predicates such as "kidnap", "explode", and "bomb". In particular, there is an unusual subject in the KG with relation to Kiev -- Jim Mattis, the United States Secretary of Defense. Without any "common sense" or "domain knowledge", it is hard to interpret the triple "Jim Mattis is in Kiev" as an activity of interests or potential anomaly. Actually, many SPO triples require background or domain knowledge to be better interpreted. To solve this problem, we again take the advantage of the more general KG such as Google Knowledge Graph to obtain existing facts or background knowledge about entities and their relations.

In Figure 3, we show an example of obtaining background knowledge from Google KG to better detect a potential anomaly. Given two triples "Jim Mattis is in Kiev" and "Petro Poroshenko is in Kiev", it is not easy to decide which activity is more unusual without any background information regarding to who are Jim Mattis and Petro Poroshenko. However, by querying Google KG, we can obtain some attributes about the two entities. Specifically, simply knowing the occupations or titles of the two persons, i.e., "Jim Mattis is the 26th and current United States Secretary of Defense and United States Marine Corps general", "Petro Poroshenko is the president of Ukraine and a Ukraine businessman and politician", we can easily make the judgment that "Petro Poroshenko is in Kiev" is more normal than "Jim Mattis is in Kiev". This shows that with the support of existing knowledge base such as Google KG, we can extract some general existing facts to help us detect potential anomalies about current activities or events. It is therefore critical to develop an automatic mechanism to collect information from other sources to validate or deny a potential anomaly.

TABLE V. ENTITY TRACKING FOR KIEV

Subject	Predicate	Object
Jim mattis	is in	Kiev
monument	vandalize	Kiev
exhibition	is in	Kiev
shooting	is in	Kiev
Kiev	regain	control
visit	is in	Kiev
Anna Kurbatova	**kidnap**	Kiev
journalist	is in	Kiev
rally	is in	Kiev
rally	is in	Kiev
car	**explode**	Kiev
wreckage	**bomb**	Kiev
car bomb	is in	Kiev
car bomb	is in	Kiev

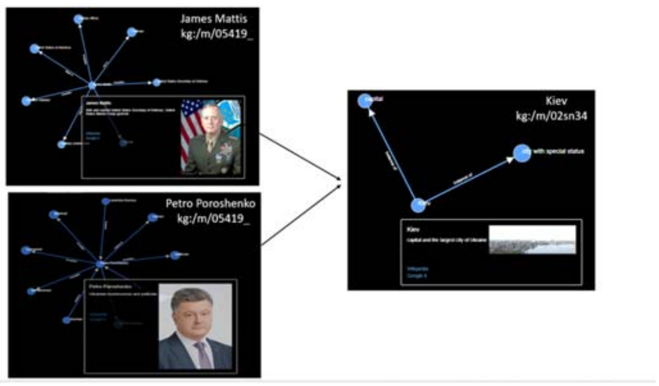


Fig. 3. Example: background information extracted from Google Knowledge Graph

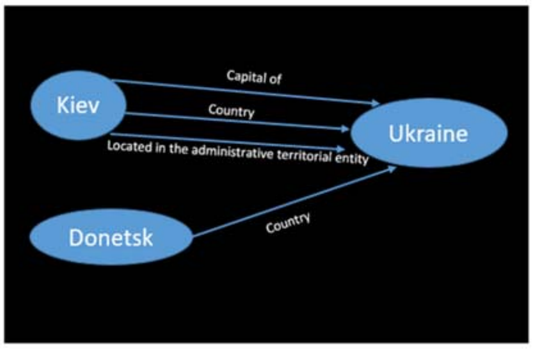


Fig. 4. Example: entity-entity relationship extracted from Wikidata

In addition to extracting basic attributes of the entities from the existing KG, we can also query the relations between two entities to verify the normalcy of their connections. For instance, "Kiev" and "Donetsk" are two of the most frequent entities in our SPO triple set. To find the reason, we can query Wikidata [15] about "Kiev" and "Ukraine", as well as "Donetsk" and "Ukraine". As shown in Figure 4, it clearly demonstrates the relations between the two pairs of entities. That is, Kiev is the capital of Ukraine, and Donetsk is a city of Ukraine. Such information provides the most basic but crucial background knowledge to understand the data set for better predictive analysis.

V. SYSTEM IMPLEMENTATION AND RESULTS

To build the data-driven pattern discovery and anomaly detection system, we use many open source tools. Apache Kafka is responsible for the streaming data delivery to different components of the system, such as the computational engine, the storage system, and the visualization system, as shown in Figure 5. The computational engine uses Lambda framework, which includes the batch processing layer using Apache Hadoop and the streaming processing layer using Apache Spark. The knowledge graph data is stored by Cassandra and visualized by D3.js.

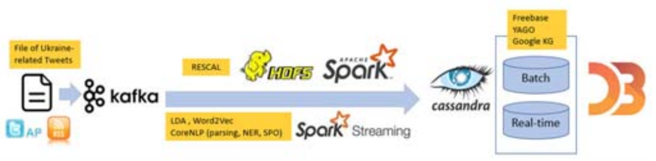


Fig. 5. Implementation Architecture

The Ukraine message stream is obtained through Live Universal Awareness Map. The link prediction method RESCAL is executed in the batch mode along with HDFS and Spark. We also keep some existing knowledge graph such as Freebase, YAGO, and Google KG in the database to support local KG construction, KG refinement, entity resolution, and relations retrieval.

Apache Spark is used as the main computational engine. It provides batch-layer processing such as link prediction using *RESCAL*. Spark Streaming processes the streaming data in a real-time fashion, such as extracting hidden topics using LDA, finding semantic-related words by Word2vec, and extracting SPO triples using Stanford CoreNLP. To show the scalability of the system, the computation time of the system (Intel Core i7 with 4 cores, 8GB RAM) with or without Apache Spark are shown in Figure 6. It can be seen that with Apache spark the CPU time increases much slower as a function of the number of messages than the one without it.

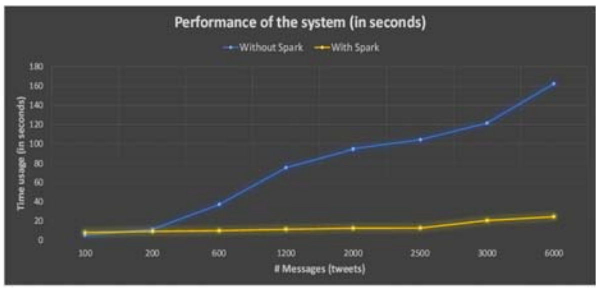


Fig. 6. Performance of System

We built a demonstration toolkit for real-time knowledge graph construction for potential anomaly detection. A snapshot of the constructed knowledge graph is shown in Figure 7. While the knowledge graph is being constructed, *RESCAL* algorithm [12] is employed to predict missing links between nodes (the red links as shown in the figure). As more nodes and edges are added, the hub node or the authority node with many incoming and outgoing edges turns from light blue to red as well.

We use a sliding-windows based method to detect potential anomalies in terms of entity, action, location and Subject-Predicate-Object (SPO) triples. The normalcy pattern is built on the knowledge graph extracted from data in the window, and the potential anomalies are detected by comparing the new data to the normalcy model. It is worth mentioning the window size can be easily adjusted to any value based on the volume and update rates of the streaming data. All entities, actions, SPOs in the window (in our current case, it means all the information within *N* consecutive days) are used to extract normal patterns. The new information (e.g., in the (*N*+1)th day) will be compared to the latest normal patterns. The new patterns and the items with large deviations from the extracted normal patterns will be regarded as potential anomalies. To evaluate the deviation between the normalcy pattern and new items, we use semantic distance for the entity and action deviation metric. As an example, with a window size of 4 days, Figure 8 (the top part) shows a screenshot of the extracted frequent entity, frequent action, and frequent SPO triplets from the messages in the time window. The corresponding potential abnormal entities, actions, and triples shown at the bottom of Figure 8 are detected based on the comparison between the streaming information and the historically extracted information. As our system processes the streaming data continually, we also extract normal location patterns from each time window and detect potential abnormal locations.

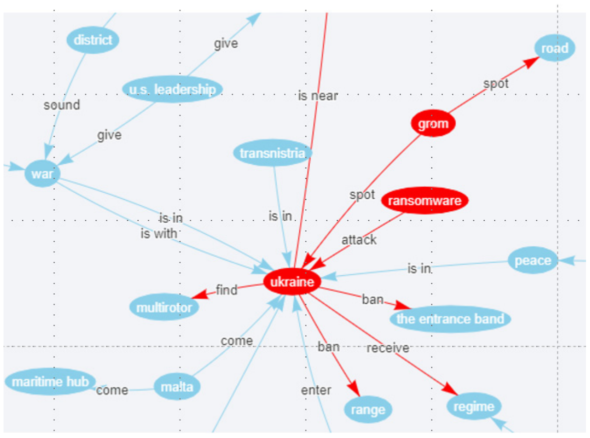


Fig. 7. A Sample Knowledge Graph

VI. Conclusion

Pattern discovery and anomaly detection is a critical element in Multi-INT fusion for surveillance and situation awareness. In this paper, we develop methodologies and implement a knowledge graph representation and reasoning system for discovering abnormal patterns and unusual activities given unstructured data received from streaming open sources. Specifically, based on the constructed knowledge graph from historical data, the normalcy model for entity, action, and triplets are established. The information of the incoming streaming data is compared to the normalcy model to detect abnormal activities. The system is implemented based on the lambda framework where the batch layer stores the static knowledge graph while the streaming layer digests the streaming information (tweets), extracts entities, and their relations. Potential anomalies are detected based on the patterns discovered through the static knowledge graph and dynamic knowledge in a sliding window. The real-time tweets data are used for test and evaluation, and the preliminary results showed scalable computation and promising performance in detecting potential abnormal pattern and activities. While the system was developed and demonstrated for a specific scenario, the methodologies and the system architecture are general and can be easily extended and applied to different domains. Some of the potential future research directions include developing an automatic mechanism to collect information from credible sources to validate or deny a potential anomaly and developing active data collection strategies based on information value to maximize the probability of correct anomaly detection.

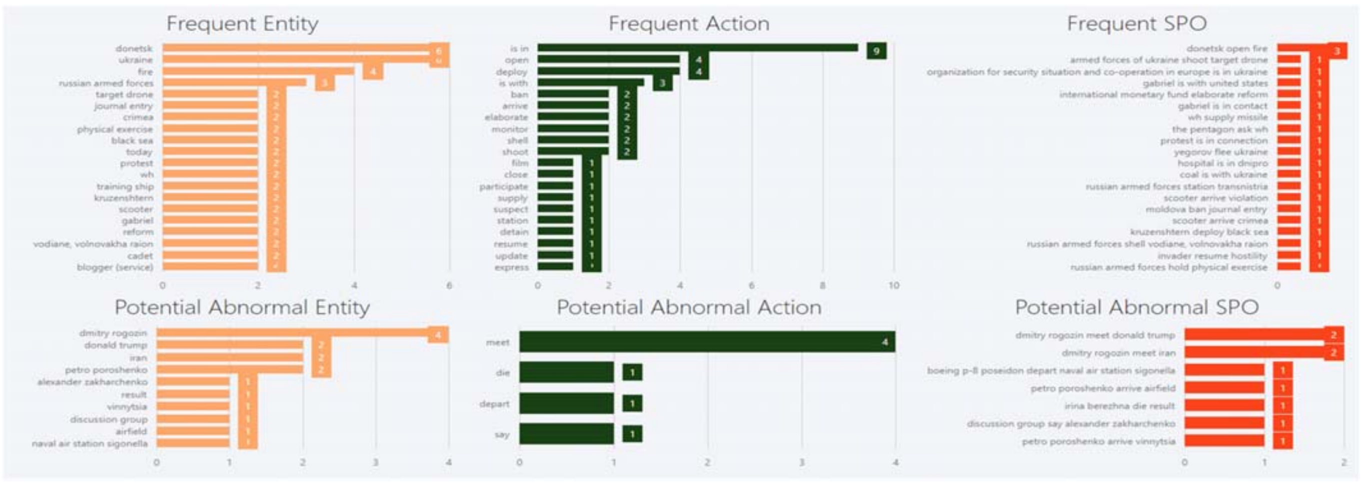


Fig. 8. Pattern Discovery and Abnormal Detection

Acknowledgment

This research was partially supported by the United States Air Force under contract number FA8750-17-C-0224. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the United States Air Force. The authors greatly appreciate Christopher Banas and Adnan Bubalo from the Air Force Research Lab for their guidance and beneficial discussions.

References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, ‘‘Freebase: A collaboratively created graph database for structuring human knowledge,’’ in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2008,pp. 1247–1250.

[2] F. M. Suchanek, G. Kasneci, and G. Weikum, ‘‘Yago: A core of semantic knowledge,’’ in Proc. 16th Int. Conf. World Wide Web, 2007, pp. 697–706.

[3] A. Carlson et al., ‘‘Toward an architecture for never-ending language learning,’’ in Proc. 24th Conf. Artif. Intell.,2010, pp. 1306–1313.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, “DBpedia: A Nucleus for a Web of Open Data,” in The Semantic Web Springer Berlin Heidelberg, 2007, vol. 4825, pp. 722–735.

[5] F. Christiane, WordNet and wordnets. In: Brown, Keith et al. (eds.), Encyclopedia of Language and Linguistics, Second Edition, Oxford: Elsevier, 2005, pp. 665-670

[6] A. Singhal, “Introducing the Knowledge Graph: things, not strings,” May 2012. [Online]. Available:http://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html

[7] M. Nickel et al. “A review of relational machine learning for knowledge graphs. Proceedings of IEEE, vol. 104, no.1, 2016, pp. 11-33

[8] G. Zhou and J. Su. Named entity recognition using an HMM-based chunk tagger. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (pp. 473-480). 2002.

[9] D. Klein et al. Named entity recognition with character-level models. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL. 2003.

[10] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL. 2003.

[11] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam, ‘‘Open information extraction: The second generation,’’ in Proc. 22nd Int. Joint Conf. Artif. Intell., Barcelona, Catalonia, Spain, 2011, vol. 1, pp. 3–10.

[12] M. Nickel, “Tensor factorization for relational learning,”Ph.D. Thesis, Ludwig-Maximilians-Universität München, Aug. 2013.

[13] J. Han, J. Pei and M. Kamber. Data mining: concepts and techniques. Elsevier. 2011.

[14] M. Christopher, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. “The Stanford CoreNLP Natural Language Processing Toolkit,” In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60. 2014.

[15] Wikipedia https://www.wikipedia.or