XLab: Early Indications & Warning from Open Source Data with Application to Biological Threat

Olga Simek, Curtis Davis, Andrew Heier, Sanjeev Mohindra, Kyle O'Brien, John Passarelli, Frederick Waugh
MIT Lincoln Laboratory, Lexington, MA
{osimek, cdavis, andrew.heier, smohindra, kyle.obrien, john.passarelli, fwaugh}@ll.mit.edu

Abstract

XLab is an early warning system that addresses a broad range of national security threats using a flexible, rapidly reconfigurable architecture. XLab enables intelligence analysts to visualize, explore, and query a knowledge base constructed from multiple data sources, guided by subject matter expertise codified in threat model graphs.

This paper describes a novel system prototype that addresses threats arising from biological weapons of mass destruction. The prototype applies knowledge extraction analytics—including link estimation, entity disambiguation, and event detection—to build a knowledge base of 40 million entities and 140 million relationships from open sources.

Exact and inexact subgraph matching analytics enable analysts to search the knowledge base for instances of modeled threats. The paper introduces new methods for inexact matching that accommodate threat models with temporal and geospatial patterns. System performance is demonstrated using several simplified threat models and an embedded scenario.

1. Introduction[*]

As a result of diverse trends in globalization, communications, and advanced technology diffusion, many nations today face a variety of security challenges from peer nations, rogue or failed states, and criminal and terrorist organizations. Many of these same trends have also vastly increased the volume of heterogeneous data that intelligence analysts must search to detect these threats. The volume of open source data in particular is expected to continue to grow exponentially, with estimates that the digital universe will grow by a factor of 10—from 4.4 trillion gigabytes to 44 trillion gigabytes—between 2013 and 2020 [1]. The broad range of possible threats, combined with the vast quantities of data that must be searched to detect them, present intelligence analysts with significant challenges.

XLab is a prototype software system that addresses these challenges by providing analysts with advanced analytics for extracting early warning signals from large volumes of heterogeneous data, built on a flexible, rapidly reconfigurable architecture. The goal of XLab is to provide analysts with the ability to visualize, explore, and query a knowledge base constructed from multiple data sources and expressed as a graph.

A key component of XLab is the ability to search this graph for instances of threat models, small graphs that describe anticipated threatening activity. This activity is often of interest not because of the characteristics of a single actor or event, but because of relationships or activity patterns among a group of actors and events. If these relationships are also expressed as graphs, then searching the knowledge base for threatening activity becomes a graph matching problem. One approach is to apply subgraph isomorphism algorithms to search the knowledge base for data that matches the threat models exactly. However, being able to find inexact matches is critical as well, because the knowledge base may be incomplete due to limited observability of threatening activity, because threat models may be incomplete due to limited analyst knowledge of threat operations, and because it enables discovery of novel threats analysts may not have even considered [2].

An initial XLab system prototype has been constructed that addresses threats associated with biological weapons of mass destruction (bio-WMDs). The prototype uses open sources to build a knowledge base of 40 million entities and 140

[*] This material is based upon work supported by the Office of the Secretary of Defense under Air Force Contract No. FA8721-05-C-0002 and/or FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of the Secretary of Defense.

million relationships that can be visualized and queried using open source tools. Knowledge extraction analytics implemented in the prototype include entity disambiguation, link estimation, and event detection. Both exact and inexact subgraph matching have been implemented for detecting threat model instances in the knowledge base. For inexact matching, a state-of-the-art algorithm was extended to ensure timely response and to accommodate threat models with temporal and geospatial patterns. The system prototype performance is demonstrated using several simplified threat models and an embedded scenario.

The paper is organized as follows: Section 2 discusses related work; Section 3 describes knowledge base construction; threat models are explained in Section 4; Sections 5 and 6 deal with exact and inexact matching, respectively; Section 7 reports on system-testing efforts; Section 8 concludes.

2. Related work

The rapidly growing amount of information available on the internet and in other digital repositories poses a serious challenge for intelligence analysts. Many questions that analysts face can be naturally formulated as graph problems [3], yet many challenges remain, ranging from graph construction, where relevant data needs to be targeted and relationships of interest inferred, to efficient ontology and threat model generation, to effective search of the knowledge graph for information of interest. In addition, these capabilities need to be implemented on a flexible architecture that can adapt to rapidly evolving threats.

A number of authors focus on detection of biological WMD attacks that have already occurred. For example, Paul et al. use hospital data to discover anthrax attacks [4], focusing on detection of spreading symptoms after the attack. They use an inexact graph matching algorithm, Truncated Search Tree (TruST) [5], to search a knowledge base for instances of a template describing anthrax inhalation symptoms. To enable TruST to scale to large data sets, various problem-specific heuristics are employed. Hu et al. create an agent-based model of a bioterrorist attack on connected cities [6] that aims to detect a bioterrorism attack before a sizeable proportion of the population is infected.

A related and challenging area is risk assessment, which typically relies on probability estimates provided by subject matter experts (SMEs). Steinberg [7] provides a general Bayesian framework and discusses in detail a threat model paradigm that takes into consideration the means, motive, and opportunity of threat actors. XLab threat models, discussed in Section 4, make extensive use of this paradigm. Koblentz [8] discusses biases inherent to risk estimation of chemical, biological, radiological or nuclear (CBRN) weapons use.

A number of authors address the graph isomorphism problem; for example, Carletti et al. [9] focus on large, dense graphs, while Babai [10] shows that graph isomorphism problem can be solved in quasi-polynomial time. By contrast, inexact graph matching is a much newer research area and the approaches can be basically broken down into two categories: index-based and index-free. NeMa [11] is an index-based algorithm that indexes the neighborhood of every node and uses an iterative inference algorithm similar to loopy belief propagation. Index-based algorithms tend to not scale to large data sets because of the enormous amount of storage space required for the index.

For our work we have selected an algorithm by Tong et al. [12] and its extension to highly attributed graphs [13] (the authors have also recently modified their approach to address iterative subgraph matching as query graph gets refined [14]). Tong's index-free algorithm allows for only the essential entities in the threat model—the ones that are required to match—to be specified. Nodes are inserted as needed for best-effort matches. This approach is attractive because, in contrast to indexed approaches, its run time scales linearly with graph size (i.e., number of nodes and edges). As described in Section 6, we extended the approach to incorporate both temporal and geospatial properties of threat models.

3. Knowledge base construction

Knowledge base construction is the process of data acquisition, information extraction, and persistence that results in a database of structured information. For XLab, the knowledge base provides the inputs to subsequent analysis such as threat detection and decision support aids. This section describes the knowledge base construction process.

In the data acquisition phase, we gather information from a variety of open sources whose subject matter is relevant to the bio-WMD threat model. Three data sources were chosen: an academic article database that provides information about individuals with bio-WMD technical expertise; an online social media platform that contributes evidence of terror group intent, recruiting, and attack planning and execution; and a commercial open-source service that provides a wide variety of documents—as well as entities, relationships, and events extracted from them—that can be mapped to many elements of the threat model.

As part of the data acquisition phase, filters are applied to ensure that data inserted into the knowledge base is relevant to the threat model while minimizing irrelevant information that can increase false alarms, hinder the discovery process, and lead to unmanageable knowledge base growth. These filters include lists of pathogens, technologies, equipment, facilities, individuals, and social media accounts known to be associated with either terror groups or bio-WMD development. For example, the Australia Group Control List of Dual-use Biological Equipment provides a list of specialized technical equipment; academic articles and textbooks [15] provide lists of government and commercial facilities associated with dual-use biotechnologies; and several websites provide lists of online accounts affiliated with terror groups.

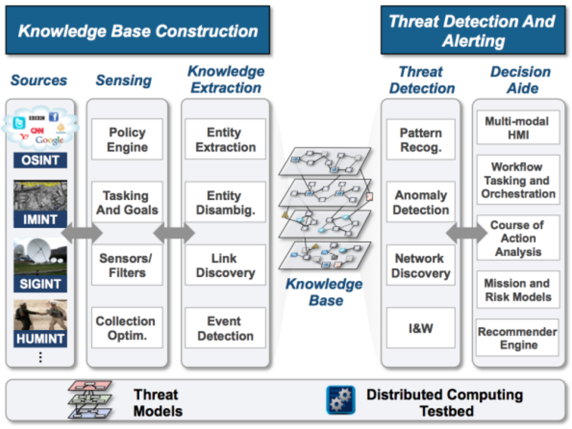


Figure 1. XLab Objective System Architecture

In the knowledge extraction phase, references to entities of interest—such as people, organizations, pathogens, locations, facilities, and social media accounts—are detected in the data and inserted into the knowledge base. A variety of extraction methods are used depending on the degree of available structure in the underlying data.

For structured datasets we derive high-confidence, exact graph representations. For example, formatted records in the academic article database can be directly parsed to yield authors, documents, and associated author/document relationships. For unstructured data sources, we use a variety of statistically trained named entity recognition (NER) models—such as the Stanford Named Entity Recognizer [16] and the MIT Information Extraction (MITIE) library [17] — which detect and label entities in text. Confidence scores produced by these methods are stored along with data object metadata to enable computation of compounded estimates of fact veracity.

Other forms of knowledge extraction implemented include entity disambiguation, link estimation, and event detection. Entity disambiguation refers to the inference either that a single text string refers to several different entities or that different strings refer to the same entity. A simple token-based string matching approach using a Jaro-Winkler distance metric [18] was implemented to resolve entities. Link estimation refers to the discovery of hidden relationships in the knowledge base. For the prototype, a simple link estimation algorithm based on *n*-gram matching was used to link real names with online account handles. For both entity disambiguation and link estimation, more sophisticated analytics incorporating social, textual, or spatiotemporal patterns of life and other features can be implemented to achieve better performance. Finally, event detection refers to the extraction of particular types of activities from their textual description. The prototype leveraged third party analytics to detect events, for example travel events of the form "person *P* traveled to location *L*" in news reports, social media posts, and other text documents.

The extracted knowledge is represented as a multi-directed property graph whose vertices represent extracted entities and whose edges represent relationships between entities. A property graph is a graph in which vertices and edges may have associated attributes; for example, a vertex representing a person may have first name, last name, and date of birth attributes. A multi-directed graph is a graph in which vertices may be connected by multiple edges, each of which may have an associated semantic direction. For instance, if a Person entity is the author of a Document entity, an edge of type "Wrote" points from the Person to the Document. Once the data is processed into this representation, it becomes possible to perform powerful queries both for single entities and for graph template patterns. Such queries are discussed further in Sections 5 and 6. A key element of the knowledge extraction process is a schema of entity and relationship types that enables data to be logically organized in a database.

In addition to the data described above, the knowledge base also incorporates data provenance information, so that derived data products and alerts have a traceable history to their original records. To accomplish this, unprocessed data is indexed separately from subsequent derived data, and each derived record has a reference to its original record. This separation also facilitates reprocessing of the original source data as information extraction algorithms improve.

From a software architecture point of view, we have incorporated a number of distributed computing technologies to enable processing and storage of large volumes of data. Figure 2 depicts the knowledge base software architecture, showing the primary elements of data processing and persistence. The architecture is built on a distributed messaging bus that accepts parsed raw data and serves as the interface for subsequent processing such as information extraction and indexing. The sensing and knowledge extraction steps described above, combined with the software architecture, yielded a prototype knowledge base containing approximately 40 million entities with 140 million relationships between them.

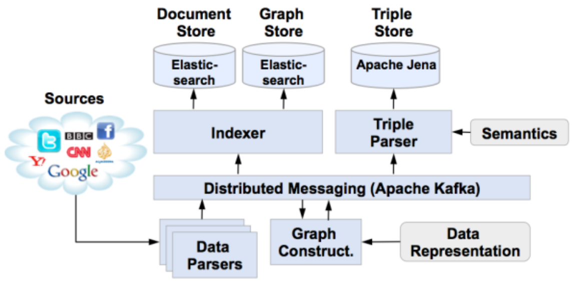


Figure 2. Graph Construction and Persistence Architecture

The volume and complexity of this data lead to visualization, exploration, and query construction and execution challenges. The XLab objective architecture of Figure 1 envisions providing intelligence analysts with tools for visualizing small regions of the knowledge base; navigating easily between regions; adding, deleting, and correcting information; constructing graph-based queries using an intuitive interface; and automatically generating prioritized lists of knowledge base entities or subgraphs that correlate with threat models. For the prototype, visualization was implemented using the open-source Gephi graph visualization tool [19], while simple query capability was achieved using ElasticSearch [20], an open-source data indexing and search tool. The next section describes how threat models are designed to enable advanced knowledge discovery and threat detection.

4. Threat models

Threat models encode the knowledge of a subject matter expert (SME) about a particular threat into a model that can be used to search the knowledge base. For the XLab bio-WMD prototype, SME threat understanding is conceptualized as a network of activity between individuals, groups, locations, and critical equipment or resources. This network is encoded as a graph that may incorporate any entity or link type defined in the knowledge base schema. Additionally, temporal and spatial constraints can be imposed across a series of entities or links to capture a sequence of related activities. Such constraints are particularly useful to reduce the number of search results for threat models with commonly occurring entities.

When constructing a threat model, SMEs can elect to model either a small element of a threat or their full threat understanding based on the means/motive/opportunity paradigm [7].

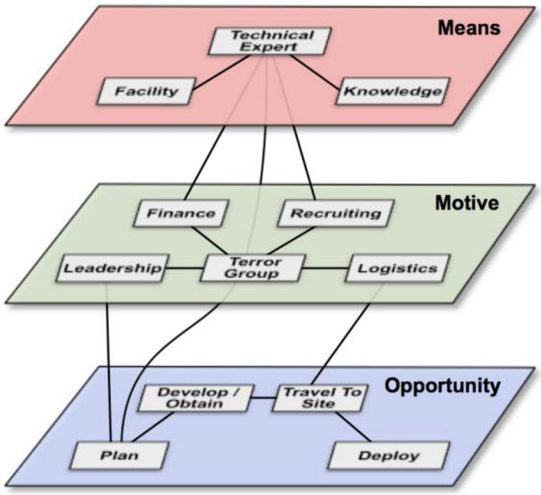


Figure 3. Bio-WMD Threat Model

Figure 3 depicts an example of a full threat model for the bio-WMD threat. In this model, a technical expert with relevant knowledge and access to a biotechnology facility (means layer) is recruited and financed by a terror group (motive layer). Together the expert and group plan an attack, develop a bio-WMD, travel to an attack site, and deploy the weapon (opportunity layer). Once a threat model is constructed, it can be used to search the knowledge base for matches that may indicate real-world threatening activity. Techniques for conducting such searches are described in the next two sections.

5. Exact graph matching

Once a knowledge base and threat model have been constructed, tools are needed for finding matches to the threat model and supporting *ad hoc* exploratory queries. Matches to a particular threat model are considered a form of the subgraph isomorphism problem [21], which is the task of finding a subgraph H of the data graph G that is isomorphic to the threat model graph or "query graph" Q. Isomorphism between graphs H and Q means that there is a one-to-one mapping or *bijection* between the two vertex sets $f: V(H) \rightarrow V(Q)$ and that any two vertices (u, v) in H are adjacent if and only if $f(u)$ and $f(v)$ are adjacent in Q. Isomorphism preserves structure. Also, H is a subgraph of G if the vertices and edges in H form a subset of the vertices and edges in G. Exact graph matching implies strict subgraph structure and attribute matches, as opposed to inexact graph matching, for which small deviations from the query structure and attributes are allowed. Section 6 is devoted to inexact graph matching.

The advantage of constructing a graph of interrelated records is that it supports more complex queries than a traditional search engine that finds keyword matches to single records stored in isolation. For example, suppose an analyst is interested in using academic citation data to uncover collaborations or expertise transfer between laboratories. With traditional search engines, the process would be laborious and many searches would be required. On the other hand, when this information is encoded as a graph of Author, Facility, and Article nodes as shown in Figure 4, the desired relationships are also encoded implicitly, and the same query is represented simply as a small graph of relationships among the entities in question, shown in Figure 5. Searching the knowledge base for isomorphic matches to this graph enables the analyst to find results quickly and automatically, as shown in the example of Figure 6. In addition to finding the pattern of interest, the analyst can also easily access rich contextual information, such as the co-authors that are related to the author in question.

Isomorphic graph matching is implemented in the XLab system using iterative graph traversals guided by a backtracking algorithm on a search tree. For a given threat model, an analyst inputs the graph as a set of directed vertex-edge-vertex triples with constraints on the vertex and edge properties. The algorithm starts by iterating over each triple and querying the database for entities matching the set of properties specified for the first vertex. These results become candidates for graph matches. Then the algorithm iterates over each candidate and traverses over the edges, filtering edges that do not meet edge property constraints, and gathers the terminal node for the triple. Each result node becomes another candidate for a complete match in a tree-like structure. The algorithm exhaustively iterates over every candidate and returns complete matches if found, or until all candidates have been searched.

We chose to persist our graph in Elasticsearch, a NoSQL document-oriented distributed database. Both graph vertices and edges are treated as atomic records in the database, where structure is encoded by storing vertex IDs in each edge record. Thus, to traverse the graph, a series of joins must be done to match vertices to their associated edges and then to their terminal vertices. This storage approach also allows us to perform powerful single-record queries with support for geographic, temporal, keyword, and field matching queries.

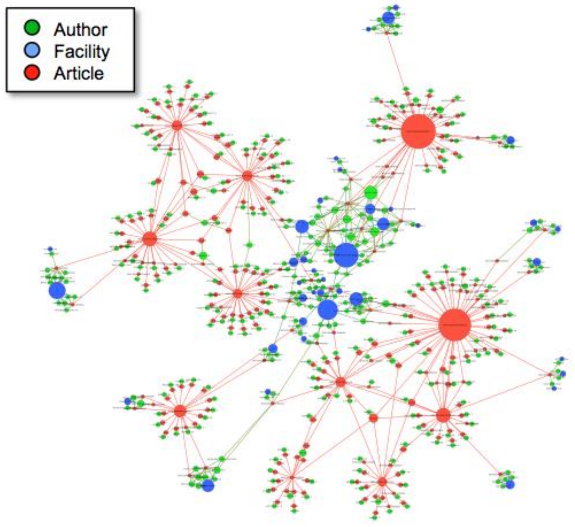


Figure 4. Relationships Extracted from Academic Article Data

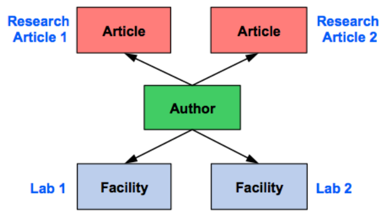


Figure 5. Example Academic Article Query

To scale the graph matching capability, the system incorporates Apache Spark, an in-memory cluster computing framework. Spark integrates with Elasticsearch through Hadoop, which allows fast ingest, and GraphFrames allows parallelized graph search through an intuitive graph pattern language [22]. Without optimization, a full-graph query consisting of two nodes (one of type Person and the other of type Organization) and an edge connecting them takes about 5 minutes to complete on an 8-core machine with 12 GB of memory.

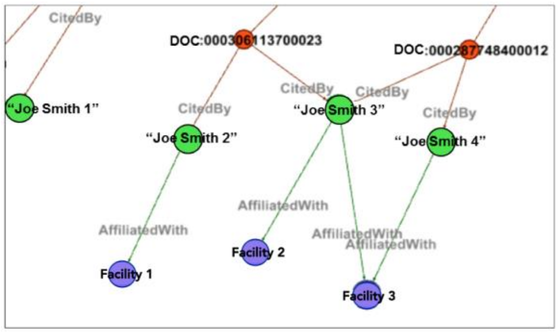


Figure 6. Exact Matching Result for Example Query

Figure 8 shows some performance metrics for various resource configurations. These results were computed on a graph of 40 million nodes and 140 million edges.

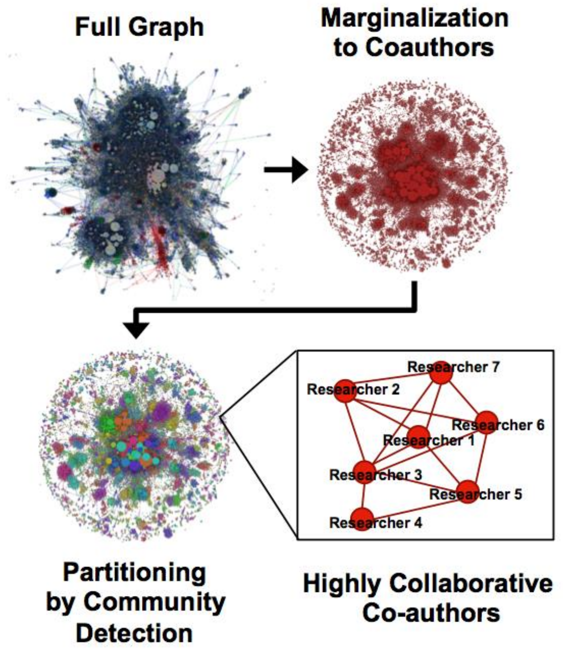
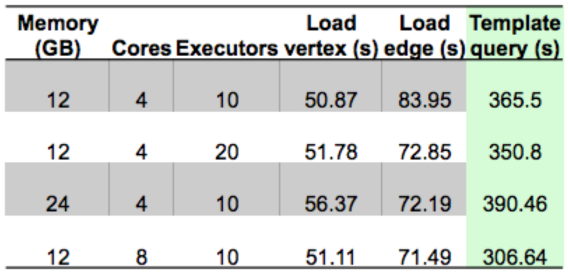

Full Graph
Marginalization to Coauthors
Researcher 7
Researcher 2
Researcher 6
Researcher 1
Researcher 3
Researcher 5
Researcher 4
Partitioning by Community Detection
Highly Collaborative Co-authors

Figure 7. Large-Scale Graph Analytics

In order to compare our computing framework to existing graph databases, we ran graph matching tests between GraphFrames and Titan, the graph database management system with Apache Cassandra as the storage backend. We persisted a smaller graph dataset of 11.8k nodes and 13.8k edges in both systems and ran the same query consisting of a moderately complex graph of 4 nodes and 4 edges. Query performance was comparable: GraphFrames averaged about 17.2 seconds while Titan averaged about 16.1 seconds. In addition to search, retrieval, and basic statistics offered by graph databases such as Titan, our computing framework allows us to easily build advanced analytics and algorithms. Figure 7 shows an example of such processing in which the full data graph has been marginalized into a graph of coauthors, labeled using a community detection algorithm, then filtered to uncover clusters of highly-collaborative researchers. All of these operations are possible as distributed computations in Spark.



Memory (GB)	Cores	Executors	Load vertex (s)	Load edge (s)	Template query (s)
12	4	10	50.87	83.95	365.5
12	4	20	51.78	72.85	350.8
24	4	10	56.37	72.19	390.46
12	8	10	51.11	71.49	306.64

Figure 8. Exact Graph Matching Run Times

6. Inexact graph matching

When threat models are only partially understood, or when discovery of novel threat configurations is of interest, inexact graph matching algorithms can be applied to calculate results of interest. For inexact matching, we adopted the structure of Tong's algorithm [12] and its extension to highly attributed graphs [13], and implemented novel approaches to improve run time and to enable expanded threat model queries to include temporal and spatial relationships.

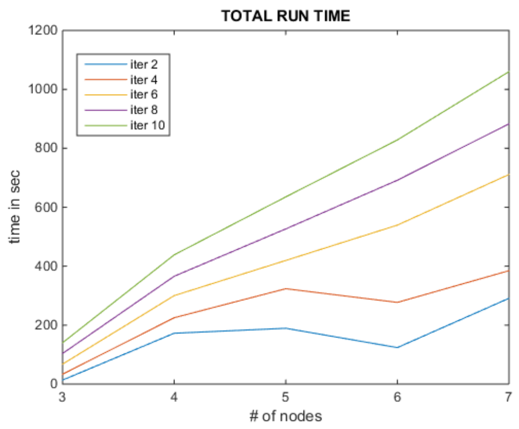


Figure 9. Run times for query graphs of varying sizes

The algorithm returns exact or best-effort subgraphs. The number of graphs returned is specified by the user. Prior to searching the graph, the algorithm calculates random walks with restart values to rank nodes based on their proximity to other nodes. Using these rankings and the given query attributes, a desirable (usually central and well-connected) seed node is selected. Neighbor nodes are similarly selected by finding the nearest highly ranked node that matches another attribute of the query graph. After selecting a neighboring node, a path is found to connect these nearby neighbors to seed nodes. This is done for all nodes in the query graph and results in a best effort sub-graph. The algorithm run time scales linearly with the size of the query graph since each node in the query graph, except for the initial node picked by the algorithm, is processed sequentially using exactly the same steps each time. Figure 9 shows run times for increasing sizes of query graphs and increasing number of iterations for random walk with restart calculations (see [12] and [13] for details). Tong et al. have demonstrated empirically that the run time scales with the number of nodes and the number of edges in the data graph as well. They have shown this on a wide variety of graph sizes (see [13] for details).

Simple model in Figure 10 seeks to uncover individuals currently working in a specific U.S. geographic area who have previously been employed by a non-U.S. biotechnology company.

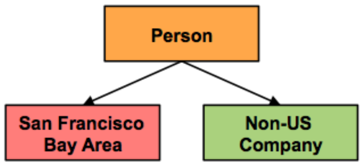


Figure 10. Threat Model

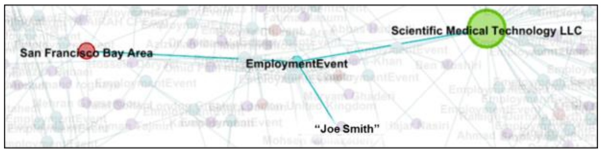


Figure 11. Inexact Graph Matching Result

The knowledge graph contains no exact matches for this threat model but does contain matches that are semantically identical, one of which is shown in Figure 11. The algorithm correctly discovers all 9 of these matches in the full knowledge graph. Recall in this instance is 1 but it should be noted that recall varies widely depending on size and structure of both the query graph and the data graph, and more investigation is needed here. It should also be noted that the goal of the algorithm is to present to the analyst reasonable matches of interest and that the algorithm can easily be combined with SQL-based methods for exact match discovery.

Inexact graph matching is also useful when an analyst knows certain connections and nodes of interest but does not have a full understanding of the various possible connections. For example, the threat model in Figure 12 describes a scientist who has published on pathogen research and who has also communicated online with an individual who is associated with an event of interest. A similar threat model is discussed further in Section 7.

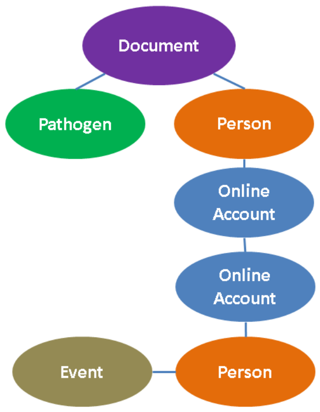


Figure 12. Threat Model

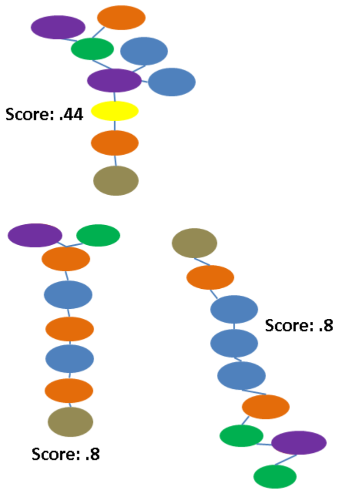


Figure 13. Inexact Graph Matching Results

Figure 13 shows some of the matches returned for the query of Figure 12. The node colors represent entity types. To evaluate how closely the resulting matches correspond to the query graph, we have created a goodness score G, defined as

$$G = \frac{-2u - e + 3c + m}{3t + q} \quad (1)$$

where

$u =$ number of unconnected nodes in result graph

$e =$ number of extra nodes in result graph not in query graph

$c =$ number of edges in result graph that connect two matching vertices

$m =$ number of nodes in result graph matching nodes in query graph

$t =$ number of edges in query graph
$q =$ number of nodes in query graph

A result graph that perfectly matches a query graph receives a score of 1. Figure 14 demonstrates calculation of G.

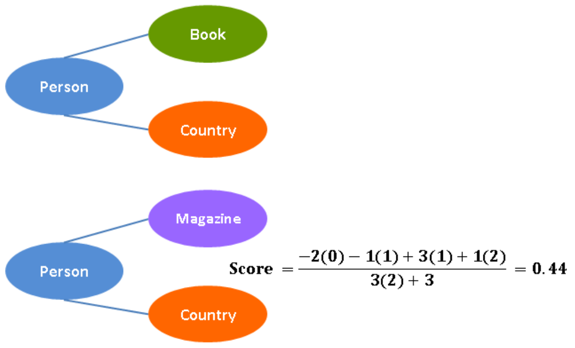

Book
Person
Country
Magazine
Person
Country
$\text{Score} = \frac{-2(0) - 1(1) + 3(1) + 1(2)}{3(2) + 3} = 0.44$

Figure 14. Sample Query (top) and Calculation of Goodness Score G (bottom)

6.1. Geospatial and temporal filters

In practice, geospatial and temporal constraints are key elements of many threat models. For instance, a threat model may contain events E_1 and E_2 such that E_1 always occurs before E_2. The knowledge graph, however, as represented for the inexact matching algorithm execution, does not facilitate searches incorporating such constraints. The storage format consists of two sparse matrices, node-to-node matrix N and node-to-attribute matrix A:

$$N = [n_i, n_j] \quad i, j = (1, \dots, n) \tag{2}$$

$$A = [n_i, a_j] \quad i = (1, \dots, n);\ j = (1, \dots, a) \tag{3}$$

where n is the number of nodes and a is the number of the attribute values. As constructed, N and A do not account for the spectrum of comparisons intrinsic to location and time. To address this deficiency, we developed two approaches, the extra nodes approach and the best rank neighbor approach.

6.1.1 Extra nodes approach. During preprocessing stage, this approach inserts an extra node between all pairs of nodes with the two attributes being compared. Figure 15 displays an example of this process for a temporal constraint. As shown in the figure, to compare the time of an employment event to the time a person is traveling, a node is inserted between all nodes of type "Employment event" and of type "Person Travel." The attributes of these additional nodes contain the difference in time or difference in longitude and latitude.

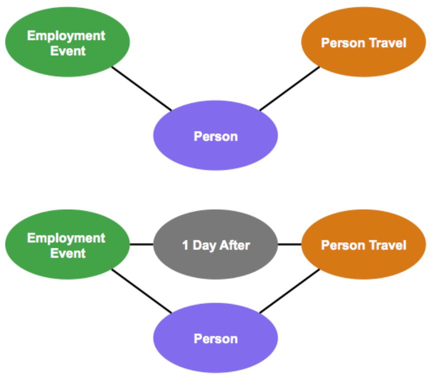


Figure 15. Query Graph (top) Augmented to Include Temporal Constraint (bottom)

The advantage of this approach is that the comparisons are embedded in the structure of the graph, so that the algorithm can continue to be used in the same way as before. The drawback is an increase in algorithm run time, arising both from the preprocessing required to augment the matrices N and A and from the added time needed to search the larger augmented matrices.

6.1.2 Best rank neighbor approach. This approach takes advantage of the algorithm's selection of high-rank neighboring nodes to select the highest rank neighboring node that also matches the given temporal or geospatial requirements. For the query depicted in Figure 16, once a specific employment event is selected, the algorithm iterates through rank-sorted neighbor nodes and ends the iteration once a neighbor node is found whose time of travel was prior to the time of the employment event.

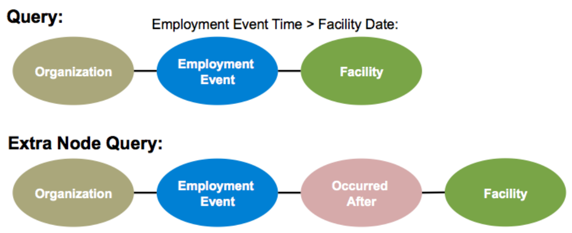


Figure 16. Queries For Run-Time Comparison

Thereby, the algorithm finds the best ranked neighbor that meets the temporal requirement. This algorithm is nearly always faster than the extra nodes approach; the additional time comes only from iterating and comparing over the given attribute.

To compare run times for these approaches, we used a representative subset of the full knowledge graph. Since both algorithms scale linearly with graph size (number of nodes and number of edges), the reported run time results can be extrapolated to estimate run times for the entire graph.

	# Nodes	# Edges	Total Time	Time: (s) 1 Result	Average Score
Extra Nodes Approach	13,855	32,059	.160	.022	.56
Best Ranked Approach	12,758	27,671	.087	.013	.65
No Temporal No Geo	12,758	27,671	.082	.012	-

Table 1. Run Time Results for Temporal Comparisons – Average of 20 Runs

Table 1 shows that, while goodness scores are similar for both methods for the query depicted in Figure 16, the average run time for the best rank neighbor approach was about half that for the extra node approach.

6.2. Graph subsampling

For some search queries, large regions of the knowledge graph may yield no matches simply because the information stored there is not relevant to the query. Graph subsampling is a technique that can improve the run time of computationally intensive graph search algorithms by prohibiting them from entering such regions.

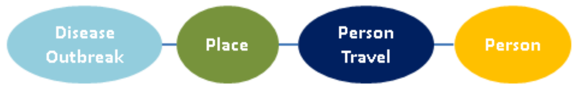


Figure 17. Query for Subsampling Approach

	# Nodes	# Edges	Total Time	Time: (s) 1 Result	Average Score
Subgraph	13,625,933	57,303,603	213.91	25.17	1
Full Graph	37,819,030	138,605,416	304.43	63.15	1

Table 2. Run Time Results for Graph Subsampling

Graph subsampling was implemented by restricting the algorithm to regions of the knowledge graph within *n* hops of any node that matches a node attribute or type in the query graph. For example, using the query in Figure 17 and choosing *n* = 2 hops to subsample the full knowledge graph cuts the number of edges and nodes that must be searched by factors of 2 and 3, respectively. Run time results for graph subsampling, averaged over 20 runs, are summarized in Table 2. Three matches were requested and three exact matches were found for both the full and subsampled graphs. The "Start to Finish Time" column lists time needed for graph subsampling plus return of three matching results. Graph subsampling is a preprocessing time cost and needs to be calculated only initially.

7. Demonstrating the system: embedded scenario

To demonstrate the capabilities of the system, we embedded a scenario within the knowledge graph based on the threat model of Figure 12. The scientist's online connection to an individual of interest involves either communication with or following an account of a radicalized person. The event of interest here is a travel event to a virology conference. We are interested in matches that show both individuals traveling to the same conference.

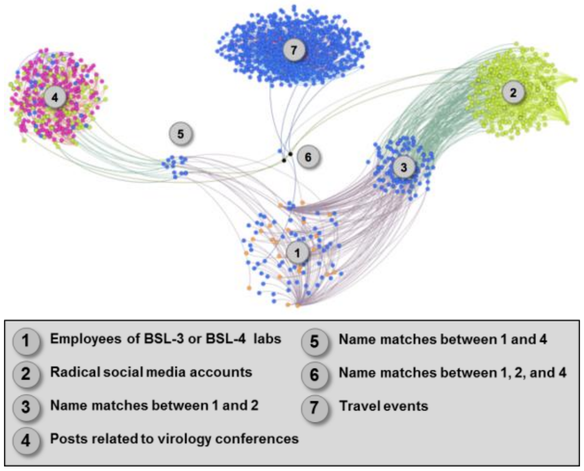

1 Employees of BSL-3 or BSL-4 labs
2 Radical social media accounts
3 Name matches between 1 and 2
4 Posts related to virology conferences
5 Name matches between 1 and 4
6 Name matches between 1, 2, and 4
7 Travel events

Figure 18. Scenario Results

Query		Knowledge Base Statistics	
1	Employees of BSL-3 or BSL-4 labs	• Total "works_at" relations:	705,416
		• "works_at" relations for BSL-3 or BSL-4 labs:	2,406
2	Radical social media accounts	• Total Social Media "follows" relations:	40,704
		• Social Media "follows" relations for ISIS accounts:	17,206
3	Name matches between 1 and 2	• Total matches:	547
4	Posts related to virology conferences	• Conference-related hashtag tweets:	21,668
5	Name matches between 1 and 4	• Total matches:	35
6	Name matches between 1, 2, and 4	• Total matches:	3
7	Travel events	• Total travel events:	76,087
		• Travel events including both person and location:	13,364

Table 3. Embedded Scenario Statistics

The exact graph matching results are shown in Figure 18. Larger clusters (1, 2, 4, and 7) represent individuals who match one of these four characteristics, while the smaller clusters (3, 5, and 6) represent individuals who match two or more characteristics. Cluster 6 contains individuals of highest interest because they possess or are linked to all criteria specified by the threat model.

Table 3 presents embedded scenario statistics, showing that the system is able to filter the number of individuals of interest from tens of thousands to three, potentially vastly reducing analyst workloads. Examples of inexact matches to a similar threat model are shown in Figure 13 in Section 6. The analyst's or subject matter expert's input would be needed to determine if those matches are of interest.

8. Conclusions and future work

This paper presents an end-to-end approach for detecting threats within large volumes of multi-modal data. A knowledge base, represented as a graph, is constructed in an automated fashion to include relevant data from disparate data sources. An approach for generating threat models that reflect a SME's threat understanding is introduced, and methods for retrieving both exact and inexact instances of the complex threat model from the knowledge base are described. We demonstrate efficient retrieval performance on a graph of 38M nodes and 140M edges and describe enhancements to the current state of the art for linearly scaling inexact subgraph matching that significantly reduce run times while maintaining high match scores. We also present new methods for discovering threat models with spatial and temporal links. Finally we show how the system can detect embedded threats within a large knowledge base. While improvements can be made to every step of the processing chain, particularly graph construction and graph matching, we believe the system provides a needed capability with sufficient speed for providing useful, timely indications and warning.

[1] IDC. The digital universe of opportunities: rich data and the increasing value of the internet of things, 2014, accessed at https://www.emc.com/leadership/digital-universe/2014iview/index.htm.

[2] G. Gross, R. Nagi, and K. Sambhoos. A fuzzy graph matching approach in intelligence analysis and maintenance of continuous situational awareness. *Information Fusion* 18, pp. 43-61, 2014.

[3] R. Colbaugh, K. Glass, and J. Gosler. Some intelligence analysis problems and their graph formulations, *J. Intelligence Community Research and Development,* 2010.

[4] J. Paul, K. Sambhoos, and G. Hariharan. Using dynamic graph matching and gravity models for early detection of bioterrorist attacks, *J. Homeland Security and Emergency Management* **6**, 2009.

[5] K. Sambhoos. Graph matching applications in high level information fusion, Ph. D. thesis, State University of New York at Buffalo, 2007.

[6] S. Hu, S. Barnes, and B. Golden. Early detection of bioterrorism: Monitoring disease using an agent-based model, *Simulation Conference (WSC)Winter*, 2014.

[7] A. Steinberg. A model for threat assessment, *Fusion Methodologies in Crisis Management*, Springer, 2015.

[8] G. Koblentz. Predicting peril or the peril of prediction? Assessing the risk of CBRN terrorism, *Terrorism and Political Violence* **23**, pp. 501–520, 2011.

[9] V. Carletti, P. Foggia, A. Sagese, and M. Vento. Introducing VF3: A New Algorithm for Subgraph Isomorphism, *International Workshop on Graph-Based Representations in Pattern Recognition*, Springer, 2017.

[10] L. Babai. Graph isomorphism in quasipolynomial time, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, 2016.

[11] A. Khan, Y. Wu, C. Aggarwal, and X. Yan. Nema: Fast graph search with label similarity, *Proc. VLDB Endowment* **6**, pp. 181–192, 2013.

[12] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs, *Proc. International Conference on Knowledge Discovery and Data Mining*, 2007.

[13] R. Pienta, A. Tamersoy, H. Tong and D. Chau. MAGE: Matching approximate patterns in richly-attributed graphs, *Big Data (Big Data), 2014 IEEE International Conference on*, 2014.

[14] B. Du, N. Cao, S. Zhang and H. Tong. FIRST: Fast Interactive Attributed Subgraph Matching, *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[15] M. Leitenberg and R. Zilinskas. *The Soviet Biological Weapons Program: A History,* Harvard University Press, 2012.

[16] Stanford Named Entity Recognizer, https://nlp.stanford.edu/software/CRF-NER.shtml.

[17] K. Geyer, K. Greenfield, A. Mensch, and O. Simek. Named entity recognition in 140 characters or less, *6th Workshop on Making Sense of Microposts, World Wide Web Conference*, pp. 78-79, 2016.

[18] T. Peng, L. Li, and J. Kennedy. A comparison of techniques for name matching. *GSTF Journal on Computing (JoC)* 2.1, 2014.

[19] M. Bastian, S. Heymann, and M. Jacomy, Gephi: an open source software for exploring and manipulating networks, *International AAAI Conference on Weblogs and Social Media*, 2009.

[20] ElasticSearch, https://www.elastic.co/quide/en/elasticsearch/reference.

[21] J. Lee and R. Kasperovics. An in-depth comparison of subgraph isomorphism algorithms in graph databases, *Proc. VLDB Endowment* **6**, pp. 133-144, 2012.

[22] D. Ankur, A. Jindal, E. Li, R. Xin, J. Gonzalez, and M. Zaharia. GraphFrames: an integrated API for mixing graph and relational queries, *Proc. Fourth International Workshop on Graph Data Management Experiences and Systems*, 2016.