Event-Centric Knowledge Graphs

Semantic Data Web Technologies Lab Report

Mst. Mahfuja Akter[1]

Informatik, Universität Bonn, Germany
s6msakte@uni-bonn.de

Abstract

The purpose of the lab project is to learn how to use Semantic Web Technologies in the context of an event centric knowledge graph and to gain experience in developing it using system design and techniques taught during the lab. One of the key requirements to facilitate semantic analytics of information regarding contemporary and historical events on the Web, in the news and in social media is the availability of reference knowledge repositories containing comprehensive representations of events and temporal relations. Existing knowledge graphs, with popular examples including DBpedia, YAGO and Wikidata, focus mostly on entity-centric information and are insufficient in terms of their coverage and completeness with respect to events and temporal relations. EventKG presented in this paper is a multilingual event-centric temporal knowledge graph that addresses this gap. EventKG incorporates over 690 thousand contemporary and historical events and over 2.3 million temporal relations extracted from several large-scale knowledge graphs and semi-structured sources and makes them available through a canonical representation. To propose a practical solution, this report focuses on the details of the integration model built as a part of our lab work. Entities from EventKG and Event registry model were retrieved and mapped together to form an integrated model containing collective information of both the models. The report further highlights the problem statement of the work detailed description of proposed approach to solve the problem including models, frameworks and dataset(s) used in the project. In the end, we summarize and give an outlook of possible future improvements..

1 Introduction

The knowledge graph represents a collection of interlinked descriptions of entities, real world objects, events, situations or abstract concepts where descriptions have a recognized construction that allows both mankind and machines to process them in a systematic and perspicuous manner. Entity descriptions contribute to one another, forming a meshwork, where each entity depict part of the description of the entities, related to it. The knowledge graph combines aspects of several data management paradigms. It can be seen as a specific type of database: because it can be queried via structured queries, graph: because it can be analysed as any other network data structure and knowledge base: because the data in it bears formal semantics, which can be used to elucidate the data and surmise new facts. Knowledge graph has semantics describing the meaning of the data. Therefore all information is self-descriptive, precise and easy to understand. It allows us to derive new information, enforce consistency constraints and the data can continuously advance by extending its scope, allows dynamic updates, data governance and manual rectification by its users. Knowledge graphs are extensively used in modern text analysis technology. Large graphs provide background knowledge and more accurate interpretation of the text. The results of the analysis are semantic tags (annotations) linking references in the text to specified concepts in the graph. These tags represent structured metadata that enables better search and further analytics. The provision

of multilingual event-centric temporal knowledge graphs such as EventKG enables structured access to representations of a large number of historical and contemporary events in a variety of language contexts. An EventKG represents changes in the world, can capture long term development and histories of entities. It is complementary to (static) encyclopaedic information found in entity centric knowledge graphs. Now, the question comes why EventKG? Policy makers and Information specialists need to know what is happening and has happened with an entity. A "database" of events would make their work easier. EventKGs current de facto standard is Google/information broker document based research and were carried out as the part of News Reader Project Jan 2013-Dec 2015(EU FP7 project). EventKG captures "who", "what", "where", "when". Deep NLP techniques can be used to extract events from large amount of texts. Thus EventKG is an extensible event-centric resource modelled in RDF. It relies on Open Data and best practices to make event data spread across different sources available through a common representation and reusable for a variety of novel algorithms and real-world applications.

2 Problem Definition

Currently, relations are spread across heterogeneous sources. First, large-scale knowledge graphs (KGs) (i.e. graph-based knowledge repositories such as Wikidata, DBpedia and YAGO) typically focus on entity-centric knowledge. Event centric information included in these sources are often not clearly identified, can be incomplete and is mostly restricted to named events and encyclopedic knowledge. Secondly, an additional source of event-centric information on the Web are proposed knowledge graphs recently which contains events from unstructured news sources using Information Extraction methods. These knowledge graphs are potentially highly noisy. The integration of events from these sources with the information in the established knowledge repositories such as DBpedia or Wikidata within a common knowledge graph does not appear meaningful. Thirdly, a popular entity such as an influential person, a city or a large organisation can impose hundreds of temporal relations within a temporal knowledge graph. For example, the entity Barack Obama possesses 2, 608 temporal relations in EventKG. Identifying the most important temporal relations within the temporal knowledge graph to provide a concise overview for a given entity that becomes a challenging task. Fourthly, there is no unanimous solution, not all the data can be fetched from the same model.

3 Architecture

Figure 1 shows, how integration have done from different data sources. At initial step, retrieve RDF data by using SPARQL endpoint[3] which is integrated for different RDF data source in JSON format. Next, retrieve event registry API data [4] using get requests in JSON. Finally integrate and fuse (based on time and location) the data to form a large ECKG dataset, maintaining the functional requirements.To extract events and sub events query integrated model using SPARQL through event name.

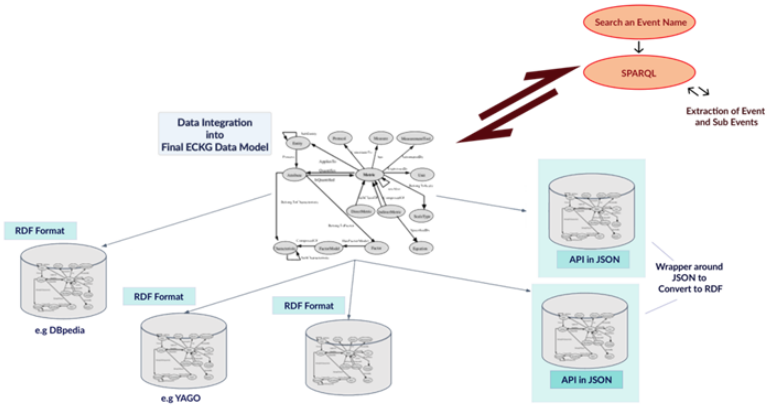

Figure 1: Architecture diagram for building ECKG data model

4 Process Pipeline

The approach developed to address the project problem has three main steps, extraction of events and entities, Integration of data and fusion of data based on user requirements. The overall process of this pipeline is to identify and extract events and entities. The goal is to then use these extractions to retrieve data from different sources and integrate into a common schema, fuse data coming from disagreeing sources and interconnect entities and events.

Figure 2 illustrates the basic approach to build event centric knowledge graph and is termed as

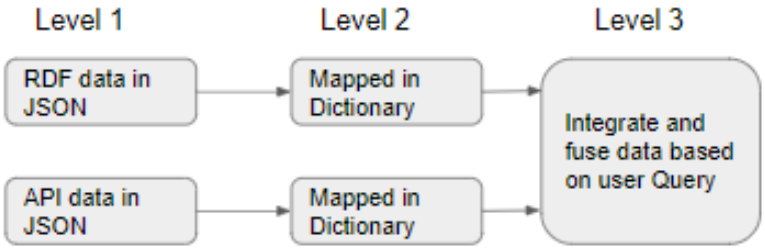

Figure 2: Basic approach for building integrated data model

bottom up approach. At level 1 two datasets are present (e.g: RDF and API) in JSON format. At level 2, mapping is done to obtain a common data model. At level 3 both the models are integrated together to build event centric knowledge graph. SPARQL endpoint is used to retrieve data from RDF dataset and eventregistry package is used to retrieve API data. Information retrieved from both the dataset is integrated together to build an event centric knowledge graph.

Figure 3 is showing the process work flow of integration model. After integration of data, output could is shown in .csv and .html

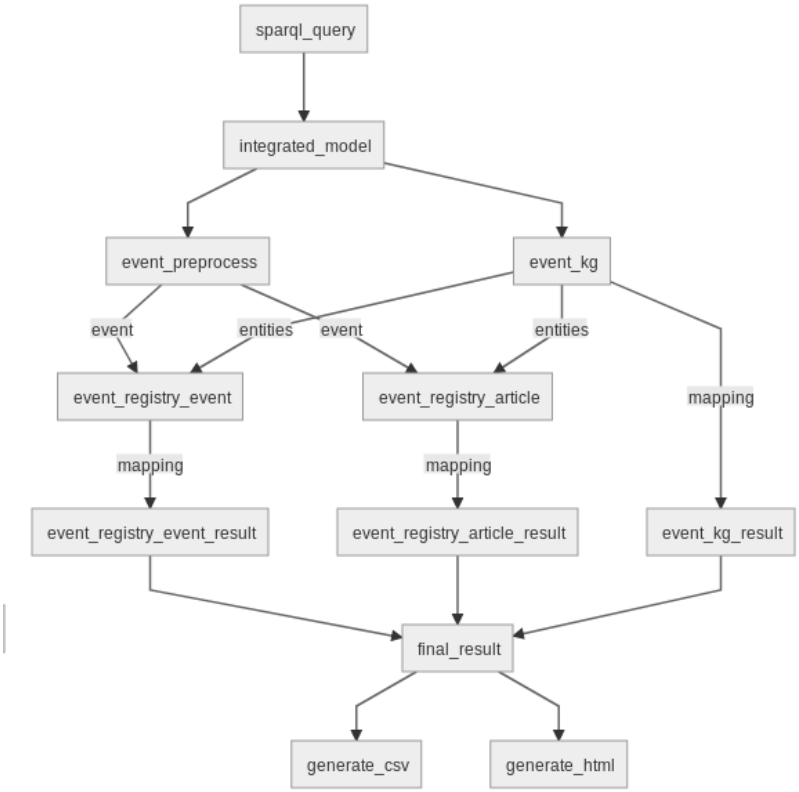

Figure 3: Work Flow of building integrated data model

5 Implementation

5.1 Event KG Dataset

The Event KG is a multilingual resource incorporating event-centric information extracted from several large-scale knowledge graphs such as Wikidata, DBpedia and YAGO, Wikipedia Current Events Portal and Wikipedia. It is an extensible event-centric resource modeled in RDF. It uses Open Data and makes event data spread across different sources available through a common representation and reusable for a variety of algorithms and applications.

5.2 Event Registry API

Event Registry is a Python package that can be used to easily access the data available in the Event Registry through the provided API. Event Registry is a system for global media monitoring. It starts by collecting news articles from over 35.000 news sources published globally in different languages. Collected articles are first annotated by identifying mentions of people, locations, organisations as well as relevant topics in them. An online clustering approach is then used to identify groups of news articles that are discussing the same event. An event for us is simply something that is being reported in several news articles.

```
Instalation:
      pip install eventregistry

Import:
      from SPARQLWrapper import SPARQLWrapper, JSON, XML

Uses:
      sparql = SPARQLWrapper("http://eventkginterface.l3s.uni-hannover.de/sparql")
      sparql.setQuery(sparql_query)
      sparql.setReturnFormat(JSON)
      results = sparql.query().convert()
```

Figure 4: EventRegistry installation and Data Extraction from EventKG

Figure 4 shows installation of eventregistry package to use API and the use of SOARQLWrapper for retrieve RDF datafrom Event KG.

5.3 Mapping from different data source

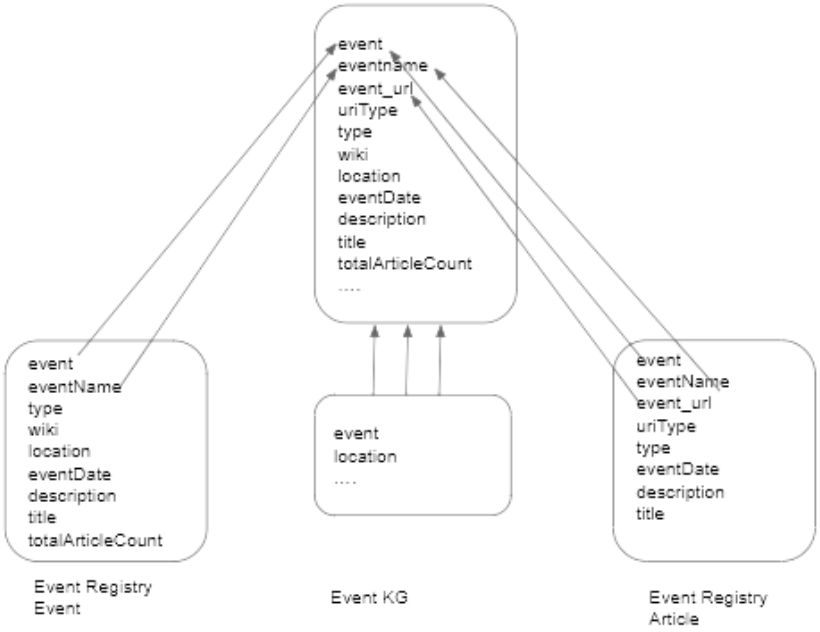

Figure 5: Mapper Function

Figure 5 shows how the entities are arranged in event registry Event And Article data. Event KG holds the all entities from RDF data source. In our integrated model, virtually it is holding all entities from both event registry (Event and Article) and RDF data entities as well. If user gives a query to retrieve those entity from any specific event, this model will provide all data from RDF and API data sources. If any of data source does not hold specific event information, integrated model will not show the data from that source.

5.4 Some sample input and respective output

```
PREFIX eventKG-s: <http://eventKG.l3s.uni-hannover.de/schema/>
PREFIX eventKG-g: <http://eventKG.l3s.uni-hannover.de/graph/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX so: <http://schema.org/>
PREFIX sem: <http://semanticweb.cs.vu.nl/2009/11/sem/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?location
WHERE
{
 ?event rdf:type sem:Event .
 ?event owl:sameAs dbr:Battle_of_Trafalgar .
 GRAPH eventKG-g:event_kg {?event sem:hasPlace ?loc} .
 GRAPH eventKG-g:dbpedia_en { ?loc owl:sameAs ?location . }
}
```

Figure 6: Example of SPARQL Query 1

location
http://dbpedia.org/resource/First_French_Empire
http://dbpedia.org/resource/Gulf_of_CÃ¡diz
http://dbpedia.org/resource/Cape_Trafalgar
http://dbpedia.org/resource/Strait_of_Gibraltar
http://dbpedia.org/resource/Atlantic_Ocean
http://en.wikipedia.org/wiki/London
http://en.wikipedia.org/wiki/Portsmouth
http://en.wikipedia.org/wiki/Spain

Figure 7: Output of SPARQL Query 1 in csv

Figure 6 shows that user gives a query for location for an event "Battle of Trafalger" and figure 7 shows the respective outcome. DBpedia link location comes from RDF data source and Wikipedia link location comes from event registry API source. Output is showing in csv format.

Figure 8 shows that user gives a query for startTime, description and event_resource for an event "World War II" and figure 9 shows the respective outcome. Some data comes from RDF data source and from event registry API source, there is no startTime and event_resource, only

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX eventKG-s: <http://eventKG.l3s.uni-hannover.de/schema/>
PREFIX eventKG-g: <http://eventKG.l3s.uni-hannover.de/graph/>
PREFIX so: <http://schema.org/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX sem: <http://semanticweb.cs.vu.nl/2009/11/sem/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?startTime ?description (?sa AS ?event_resource)
WHERE {
 ?event owl:sameAs dbr:World_War_II .
 ?event sem:hasSubEvent* ?subEvent .
 ?subEvent sem:hasBeginTimeStamp ?startTime .

 OPTIONAL {
  GRAPH eventKG-g:wikipedia_en { ?subEvent dcterms:description ?description . }
  FILTER(LANGMATCHES(LANG(?description), "en")) .
 }

 OPTIONAL { GRAPH eventKG-g:dbpedia_en { ?subEvent owl:sameAs ?sa . } . } .
 FILTER (?startTime >= "1941-02-12"^^xsd:date) .
 FILTER (?startTime < "1941-02-28"^^xsd:date) .

 FILTER(BOUND(?sa) || BOUND(?description)) .
}
GROUP BY ?startTime ?subEvent ?sa
ORDER BY ASC(?startTime)
```

Figure 8: Example of SPARQL Query 2

startTime	description	event_resource
1941-02-12	Erwin Rommel arrives in Tripoli.	
1941-02-14	Admiral Kichisaburō Nomura begins his duties as Japanese Ambassador to the United States.	
1941-02-14	World War II – Attack on Pearl Harbor: Admiral Kichisaburō Nomura begins his duties as Japanese ambassador to the United States.	
1941-02-17	The Battle of Trebeshina or the Battle of Mal Trebeshinë, was a series of engagements fought between the Greek and Italian armies in south-eastern Albania during the Greco-Italian War.	http://dbpedia.org/resource/Battle_of_Trebeshina
1941-02-19	Three Nights' Blitz over Swansea, South Wales: Over these 3 nights of intensive bombing, which lasts a total of 13 hours and 48 minutes, Swansea's town centre is almost completely obliterated by the 896 high explosive bombs employed by the Luftwaffe; 397 casualties and 230 deaths are reported.	
1941-02-22	Bombards Barawa, on the coast between Kismayo and Mogadishu.	
1941-02-25	Operation Abstention was a code name given to a British invasion of the Italian island of Kastelorizo off the Turkish Aegean coast, during the Second World War, in late February 1941.	http://dbpedia.org/resource/Operation_Abstention
1941-02-27	The Action of 27 February 1941 was a single ship action between a New Zealand cruiser and an Italian auxiliary cruiser.	http://dbpedia.org/resource/Action_of_27_February_1941
1941-02-27	The New Zealand Division cruiser HMS Leander (1931) sinks Italian armed merchant raider Ramb I off the Maldives.	
	A 93-year-old World War II veteran has reunited with his wartime sweetheart more than 70 years after they last said goodbye. Norwood Thomas was a 21-year-old paratrooper when he met 17-year-old British girl Joyce Morris in London just before D-Day. They dated for a few months before the war intervened and saw Mr Thomas sent to Normandy. After the war ended he returned to his native America but the pair wrote letters to each other, Mr Thomas even asking his beloved to come to the US and marry h.	
	A 93-year-old World War II veteran has reunited with his wartime sweetheart more than 70 years after they last said goodbye. Norwood Thomas was a 21-year-old paratrooper when he met 17-year-old British girl Joyce Morris in London just before D-Day. They dated for a few months before the war intervened and saw Mr Thomas sent to Normandy. After the war ended he returned to his native America but the pair wrote letters to each other, Mr Thomas even asking his beloved to come to the US and marry h..	

Figure 9: Output of SPARQL Query 2 in html

description of an event comes from event registry. Output data is shown in html format.

Figure 10 shows that user gives a query for event and location for an event "Brexit" and figure 11 shows the respective outcome. First data comes from RDF data source, next two data comes from event registry Event API source and last few entries comes from event registry Article. There is no location information in Article data. Output is shown in html format.

```
PREFIX eventKG-s: <http://eventKG.l3s.uni-hannover.de/schema/>
PREFIX eventKG-g: <http://eventKG.l3s.uni-hannover.de/graph/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX so: <http://schema.org/>
PREFIX sem: <http://semanticweb.cs.vu.nl/2009/11/sem/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?event, ?location
WHERE
{
 ?event rdf:type sem:Event .
 ?event owl:sameAs dbr:Brexit .
 GRAPH eventKG-g:event_kg {?event sem:hasPlace ?loc} .
 GRAPH eventKG-g:dbpedia_en { ?loc owl:sameAs ?location . }
}
```

Figure 10: Example of SPARQL Query 3

event	location
http://eventKG.l3s.uni-hannover.de/resource/event_168468	http://dbpedia.org/resource/United_Kingdom
eng-5464719	http://en.wikipedia.org/wiki/United_Kingdom
eng-5464719	http://en.wikipedia.org/wiki/European_Union
1424879140	
eng-3288539	
eng-1810234	

Figure 11: Output of SPARQL Query 3 in html

6 Conclusion

In our integrated Event KG model we are accumulating data from different RDF data source(e.g: DBpedia, Yago, etc.) and two different Event Registry data source(Event, Article) where data is based on 4 major components as WHO, WHAT, WHEN and WHERE of a particular event. We have used regular expression to extract Event from SPARQL query, search entities from eventKG output. However, those search entities are always user variant, this might have any meaningless entity name. In such case this model will give only eventKG output from RDF data source. It will skip event registry data. The future implementation can be finding the respective relational properties for respective search entities and apply this to retrieve data from event registry. Also this virtual model can be extended by a fixed data model consisting all properties from RDF and API data source.

References

[1] Simon Gottschalk, Elena Demidova, "EventKG: A Multilingual Event-CentricTemporal Knowledge Graph," *Proceedings of the Extended Semantic Web Conference (ESWC 2018)*, DOI:10.1007/978-3-319-93417-4_18, arXiv:1804.04526v1, Apr 2018.

[2] Simon Gottschalk, Elena Demidova, "EventKG: A Multilingual Event-Centric Temporal Knowledge Graph," `http://eventkg.l3s.uni-hannover.de` .

[3] Event KG SPARQL endpoint, `http://eventkginterface.l3s.uni-hannover.de/sparql`.

[4] Event Registry API, `http://eventregistry.org`.