Tools and Infrastructure for Supporting Enterprise Knowledge Graphs

Sumit Bhatia, Nidhi Rajshree, Anshu Jain, and Nitish Aggarwal

IBM Research
sumitbhatia@in.ibm.com,
{nidhi.rajshree,anshu.n.jain}@us.ibm.com,nitish.aggarwal@ibm.com

Abstract. We demonstrate EKG, a collection of tools and back-end infrastructure for creating custom, domain specific knowledge graphs. The toolkit is geared toward enterprises and government organizations where domain specific knowledge graphs are often not available. During the demo, audience members will be able to ingest their own documents and instantiate their own knowledge graphs and update them in real time. We will also present a demo app built using the toolkit consisting of more than 30 million entities and 192 million edges in order to demonstrate the kind of applications that could be built using the proposed toolkit. The app can be used to answer questions like *who are the relevant persons named Steve in context of apple computers?,* or *who are the most important persons related to Barack Obama in context of healthcare reforms act?* The functionalities of the toolkit are also exposed through REST APIs making it easier for developers to use the capabilities in their own applications.

1 Introduction

Semantic Knowledge Bases such as Knowledge Graphs play a crucial role in modern day data and knowledge management applications by offering a consolidated, concise view of the knowledge present in diverse, and often unstructured data sources. While publicly available Knowledge bases such as DBPedia [1], Freebase [4], Yago [7], etc. have been used for various applications, they represent a generic view of the World and are not suitable for developing solutions to problems often encountered by enterprises and government organizations. For example, a pharmaceutical company may require a Knowledge Base representing interactions between gene, proteins, and drugs for accelerating drug development [6]. Development of such domain-specific enterprise knowledge graphs is expensive, time consuming and requires significant human efforts and domain expertise on part of developers. Further, enterprises often have to deal with a continuous incoming stream of new data and it is crucial that the underlying knowledge graphs are constantly updated to reflect this changing knowledge. In this demonstration, we demonstrate **EKG** – a suite of tools and back-end infrastructure to assist in development of such **E**nterprise **K**nowledge **G**raphs. The proposed set of tools offers organizations *(i)* tools to extract entities and relationships of interest from unstructured documents using machine learning and rule based extractors;*(ii)* back-end infrastructure to store the extracted knowledge graph and associated metadata; *(iii)* mechanism to incrementally add to and update the graph as new documents become available; and *(iv)*

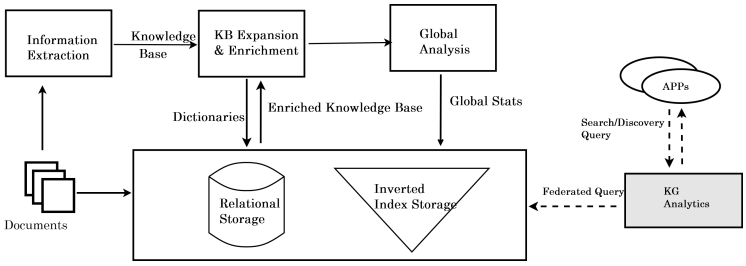


Fig. 1: System architecture

API based access to a collection of analytic methods to query and retrieve the knowledge graph.

2 System Architecture

The major components of the proposed system are illustrated in Figure 1 and are described in more detail in the following subsections.

2.1 Information Extraction Module

The input to to the system is a collection of unstructured text documents that are processed via an information extraction pipeline that extracts entities and relationships between the entities. For this extraction, the system uses SIRE (**S**tatistical **I**nformation and **R**elation **E**xtraction) toolkit [5] that deploys a Maximum Entropy classifier using lexical, syntactic, and semantic features to extract entities and relationships from text. SIRE has been successfully used for information extraction in multiple domains such as news domain, healthcare, etc. Further, we also use SystemT [1], a declarative rules based information extraction system where the end-users can specify custom information extraction rules suitable for their domain of interest by using SystemT's Annotation Query Language (AQL).

2.2 Knowledge Expansion and Enrichment Pipeline

The baseline knowledge graph output by the information extraction pipeline is then passed through the following post-processing operations in the expansion and enrichment pipeline.

Noise Removal: Since the entities and relationships are extracted automatically using trained classifiers and rules specified by SystemT, often there is some noise in the output such as entity names containing punctuation marks or other special characters due to errors in extraction (*Barack Obama;, Steve Jobs-*, etc.) Such errors are corrected using a set of replacement filters that use regular expressions to find and replace such errors. The system provides a set of about 800 filters for this task and users can either chose

[1] `https://en.wikipedia.org/wiki/IBM_SystemT`

from them or specify custom regular expressions to suite their task.

Entity Normalization: It may happen that the same entity is mentioned by different synonymous surface forms in different documents that causes the same entity to be represented as different entities corresponding to different surface forms in the graph. For example, *Barack Obama*, *Barack H. Obama*, and *Barack Hussain Obama* can be represented as three different entities in the graph. To overcome this problem, we use a dictionary of synonyms that maps different surface forms of an entity to its canonical form. A dictionary created out of dbPedia redirects is provided with the EKG toolkit, however, users can also specify custom domain specific dictionaries to use for normalization, if available.

Expansion: In many cases, it happens that the trained classifier is able to extract entities from a piece of text but the relationship between these entities could not be determined with significant confidence. For example, consider the following sentence where the extractors were able to correctly identify *Obama* and *Hillary Clinton* as two entities, but failed to identify the relationship between them.

...In 2008, Obama was nominated for president, a year after his campaign began, and after a close primary campaign against Hillary Clinton...

In such cases, the system creates a *colocation* relationship between the two identified entities and adds it to the graph. This expansion step is turned on by default to minimize the loss of useful information, however, users can turn this feature off if so desired.

2.3 Global Analysis

The enriched and expanded graph is then analyzed and several frequently required important statistics such as edge counts (number of times a relationship was observed), number of documents in which an entity or relationship was observed, etc. are computed. Further, we also compute a context model for each node (entity) in the graph. The context for an entity e consists of a language model approximated by all the terms surrounding the mentions of e in the text as well as a term vector consisting of all entity names connected with e in the graph. This context model provides an approximate representation of the *context* in which the entity e appears in the corpus. This context model is used by the analytics engine as described in Section 2.5.

2.4 Storage Back-end

The enriched and expanded graph along with pre-computed statistics and context models is then stored in a federated backend consisting of a *(i)* relational database to store the entity relationship tuples, optional domain-specific dictionaries provided by the user, and the global statistics as described above; and *(ii)* a SOLR[2] store to store the input text corpus and the context models for entities.

[2] `http://lucene.apache.org/solr/`

2.5 Analytics Engine

In addition to offering tools and infrastructure for creating and storing knowledge graphs, the EKG suite also offers a collection of methods for querying the graph for exploring entities and their relationships. Following are the major query APIs offered by the EKG toolkit.

Context Sensitive Entity Search: Given a query string and a few context terms, the API returns a ranked list of entities relevant in the given context. Due to the space constraints, we refer the reader to our previous work [3] where the probabilistic model for ranking entities in a given context has been described in detail. Further, the screenshots in Figure 2 illustrate the different results produced with changing context for the same input token *larry*.

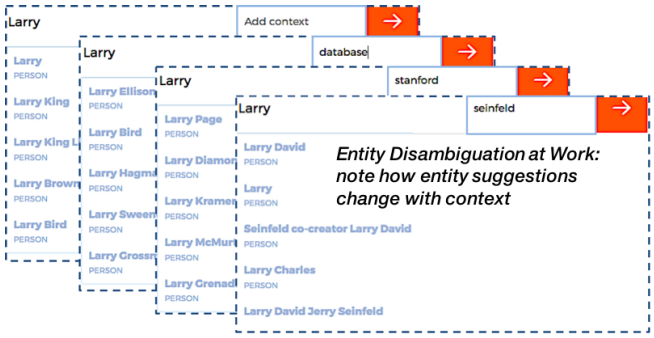
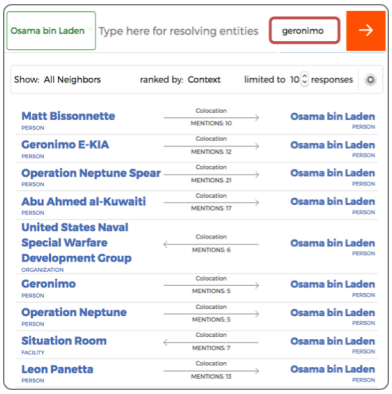


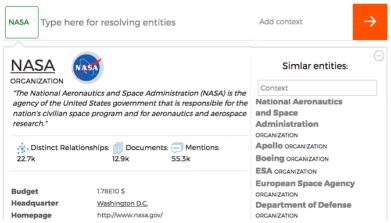
Fig. 2: Ranked list of relevant entities shown to the user under different contexts for the input token *larry*.

Relationship Search and Ranking: This API can be used to search for relationship information about one or more entities and to explore connections between different entities. The API offers different relationship search and ranking strategies to enable more fine-tuned entity exploration. For example, users can search for relationships of *Osama Bin laden* ranked by popularity, or they can perform a context sensitive relationship search for *Osama Bin laden* in context of *geronimo*. Figures **??** to 3a illustrate these examples. As can be noted in Figure 3c, the users can also query and retrieve the supporting evidence for selected relationships. For details of the relationship search algorithms, we direct the reader to our previous work [2].

Similar Entity Search: This API allows users to search for entities similar to a given entity and thus enables them to further their exploration. We use the context models for entities as described above to find entities that are observed in similar contexts in the input text corpus to determine similarity between two entities. Figure 3b illustrates the working of similar entity search with *NASA* as the input entity.



(a) Example of Context Sensitive Relationship Search



(b) Illustration of Similar Entity Search

(c) Supporting passages presented as context for selected relationships

Fig. 3: Screenshots illustrating different functionalities of EKG toolkit.

2.6 EKG REST APIs

All the operations that have been described above are exposed to the end-user through RESTful web service enabling end-users to ingest their own documents on demand, extract entities and relationships, create and update the knowledge graph, and query the graph using the analytics APIs. The organizations can thus set up their own custom knowledge graphs and incorporate this domain specific knowledge in multiple data and knowledge management applications, thereby reducing cost and complexity of enterprise knowledge management.

3 Demonstration Set Up and Requirements

We will demonstrate the capabilities of the EKG toolkit by means of a web service running on our organization's servers. We will only require a working internet connection to connect to our organization's VPN network to access the web service. Our demonstration will consist of two parts. In the first part, members of the audience will be able to ingest their own documents (text from a web page, or a passage typed by them) and create their own instances of knowledge graph representing knowledge extracted from

their documents. We will show how the system is able to handle this on demand ingestion of documents and how the statistics and state of the graph is updated in real time, a crucial requirement for various organizations. Further, users will be able to play with all the functionalities like adding custom filters (Section 2.2) for data cleaning, and query APIs to see how to query the graph. For the second part, we will present a demo app using a knowledge graph constructed from text of all articles in Wikipedia. This graph contains more than 30 millions entities and 192 million distinct relationships in comparison to 4.5 million entities and 70 million relationships in DBpedia. Using this app, users will be able to query the graph and search for entities, relationships, and play with different search and ranking strategies as described in Section 2.5. Figures 2 to 7 provide some screenshots of this demo app illustrating different functionalities offered by the EKG toolkit. The purpose of this app is to convey the capability of the proposed system to scale to graphs with tens of millions of nodes and edges, and thus, its suitability for organizations working with large amounts of data. Further, while the REST APIs are more geared towards app developers, a demo app also offers audience a visual means to explore and experiment with different functionalities offered by the proposed system.

4 Conclusion

We proposed to demonstrate the EKG toolkit that enables end-users to ingest their data, extract entities and relationships, and create knowledge graphs that could then be used in multiple knowledge management applications. Users can create custom annotators and can specify rules for information extraction using AQL and can also specify different post-processing rules for improving the quality of information extraction output. The proposed architecture is also optimized to handle incremental ingestion of data and offers a suite of query APIs that different apps can use to retrieve relevant information from the graph.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735. Springer (2007)
2. Bhatia, S., Goel, A., Bowen, E., Jain, A.: Separating wheat from the chaff - A relationship ranking algorithm. In: The Semantic Web - ESWC 2016 Satellite Events. pp. 79–83 (2016)
3. Bhatia, S., Jain, A.: Context sensitive entity linking of search queries in enterprise knowledge graphs. In: The Semantic Web - ESWC 2016 Satellite Events. pp. 50–54 (2016)
4. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD. pp. 1247–1250 (2008)
5. Castelli, V., Raghavan, H., Florian, R., Han, D.J., Luo, X., Roukos, S.: Distilling and exploring nuggets from a corpus. In: SIGIR. pp. 1006–1006 (2012)
6. Nagarajan, M., et al.: Predicting future scientific discoveries based on a networked analysis of the past literature. In: KDD. pp. 2019–2028. KDD '15 (2015)
7. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th international conference on World Wide Web. pp. 697–706. ACM (2007)