Enabling Contextualized Ontology Modeling with a Collaborative Multi-View System

Elton Soares, Raphael Thiago, Leonardo Azevedo, Sandro Fiorini, and Marcio Moreno

IBM Research, Brazil, Pasteur Ave, 146, Rio de Janeiro - RJ, Brazil
{eltons,sandro.fiorini}@ibm.com,{mmoreno,lga,raphaelt}@br.ibm.com

Abstract. Ontology modeling is an essential task for developing symbolic AI in which domain experts or knowledge engineers define taxonomies of concepts and semantic relationships to represent a particular domain's knowledge. The effective integration of symbolic knowledge with non-symbolic content enables richer knowledge representation and reasoning, and more explainable and efficient training of machine learning models used across multiple AI applications. This demo's main goal is to present how the Knowledge Explorer System (KES) multi-view architecture enables contextualized ontology modeling over a hybrid knowledge representation by making it easier to create and visualize taxonomies and relationships defined in a specific context. KES ontology modeling view supports efficient modeling of contextualized ontologies while its structural view supports their exploration, curation, and integration with non-symbolic content.

Keywords: Hyperknowledge; Hybrid Knowledge Representation; Knowledge Management; Knowledge Visualization; Knowledge Curation

1 Introduction

Ontologies have been at the center of many AI projects in recent years. As AI solutions' scale and pervasiveness grow, expressivity and algorithmic efficiency are still very relevant problems for knowledge bases. One important avenue for improvement is the effective representation of contextual knowledge.

It has become evident that current ontology modeling approaches, such as those based on W3C Semantic Web Standards[2] (WSWS), cannot address the fact that a large part of the domain knowledge represented through ontologies can only be assumed to hold under specific contexts [6]. These contexts can be related to, for example, time, space, political and cultural dimensions.

While RDF datasets provide some support for contextualization, their interpretation is ambiguous[3]. Also, there is no standard way to define explicit subgraph relationships in RDF datasets.

[2] https://www.w3.org/standards/semanticweb
[3] https://www.w3.org/TR/rdf11-datasets

As the WSWS does not explicitly prescribe an approach to model ontologies considering the contextual aspects of knowledge, this information is often embedded into ontology identifiers, entity labels, or annotations properties [6].

So, an explicit ontology modeling approach considering context is a significant issue, as the lack of widely adopted standards can cause inconsistencies in the modeling approaches utilized by different organizations or between modelers from the same organization. Besides, the explicit representation of contexts can enable more efficient reasoning and query processing [1] by allowing the reduction of the scope of a query or inference to a specific context.

The hybrid knowledge representation used by KES, namely Hyperknowledge (HK), addresses this problem. Besides, it supports integrating high-level concepts represented within the ontologies with corresponding non-symbolic content (*i.e.*, multimodal content such as machine learning (ML) models, images, videos, text, audio, etc.) [3]. HK bridges the semantic gap [7] between what a non-symbolic content means and its symbolic conceptual representation by providing first-class constructs that can be used to describe relationships between these two type of entities. HK relationships are specified by binding concepts and n-dimensional fragments of non-symbolic content [4]. It also provides constructs that enable the explicit contextualization of rich relationships between ontological terms and non-symbolic content [3].

Using KES structural and ontology views, users can collaboratively model a richer representation of the domain knowledge, which would not be possible with solutions based on purely symbolic knowledge representations.

2 Collaborative Multi-View System

KES is the first graphical system to enable the creation, exploration, and curation of HK specifications and the ingestion of multimodal, contextualized knowledge graphs generated using the HK representation or another WSWS compatible representation, in the HKBase[4] [2]. Its main focus is to assist end-users on understanding and managing HK specifications and its flexible architecture allows different visualizations of the same specification [3].

In Figure 1, we depict an overview of KES architecture and highlight the main internal components of its backend, that serves multiple frontend clients running within the user's web browsers. We also highlight the frontend components organized into two main categories: `Core` services and `Tools`. Figure 1 highlights two of these tools (*marked with eyes*): the `Renderer`, which presents the Hyperknolwedge specification as a graph and is mainly associated with the structural view; and the `Concept Hierarchy` tool, which enables users to visualize the same specification as a taxonomy of concepts.

KES Backend's main functionality is provided by six internal components. (1) `Model Service` provides an API to query and store the hybrid models in the

[4] HKBase is a middleware responsible to provide persistent storage of the HK graph.

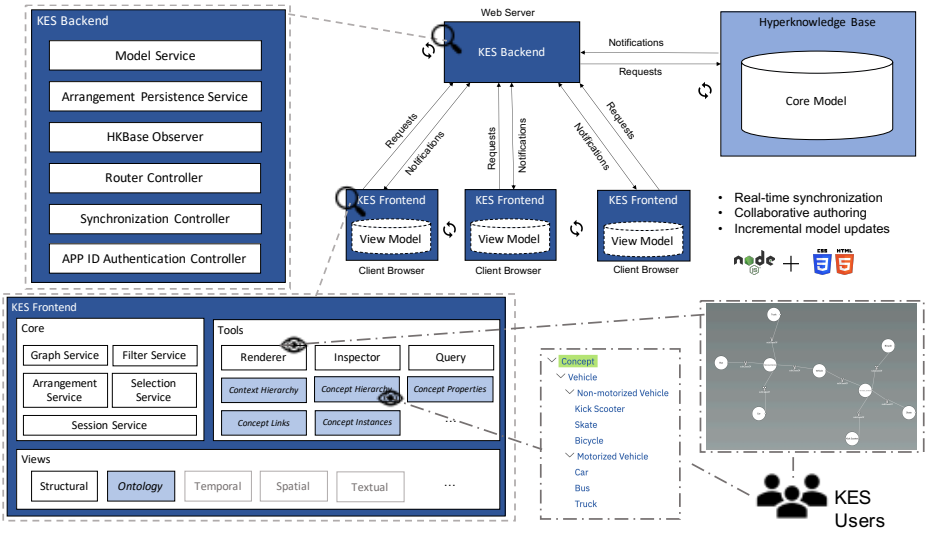

Fig. 1. KES architectural overview, including highlight of backend and frontend internal components, and two frontend Tools snapshots: `Renderer` and `Concept Hierarchy`.

HKBase (Hyperknowledge Base) using one of its storage options[5]. Model Service uses a data source (HKDatasource) implementation from *hklib*[6], the official HK library for Javascript, to connect to HKBase and propagates change notifications using an Observer pattern implementation provided by the *ninja-util*[7] library. (2) `Arrangement Persistence Service` stores the visual arrangement of entities in a 2D plane and also uses the HKDatasource to connect to HKBase and persist entities' positions in a specialized repository. (3) `HKBase Observer` registers the server as an Observer of HKBase events using either a REST or a Message Broker notification mechanism. (4) `Router Controller` provides a proxy to HKBase endpoints for the frontend, and removes the need of exposing HKBase externally just for using KES while also allowing for an additional layer of authentication and encryption. (5) `Synchronization Controller` listens to all clients event notifications, and broadcasts them to all other clients using WebSockets message protocol[8]. (6) `APP ID Authentication Controller`. provides user authentication via IBM Cloud APP ID[9], which supports multiple authentication options such as Single Sign-on (SSO) and SAML 2.0. The use of this component is optional and can be configured during deployment.

[5] HKBase currently allows using MongoDB, JanusGraph, and Apache Jena for storing symbolic content and any AWS S3 compatible object store for non-symbolic content.
[6] https://github.com/ibm-hyperknowledge/hklib
[7] https://github.com/ibm-hyperknowledge/ninja-util
[8] https://tools.ietf.org/html/rfc6455
[9] https://www.ibm.com/cloud/app-id

KES Frontend's Core services implement functionalities that are reused by multiple Tools. The main services are: (1) `Graph Service`: Provides an API to manage the HK models' local copy and synchronize them with HKBase. It uses a progressive loading strategy to optimize the navigation of large HK graphs; (2) `Arrangement Service`: Implements an automatic 2D arrangement algorithm for HK entities and provides an API to manage and notify position updates; (3) `Filter Service`: Enables filtering entities by IDs/IRIs using regex patterns; (4) `Selection Service`: Manages the users' selection state and notifies its changes. (5) `Session Service`: Handles the communication with KES Backend, notifying updates made in the local model to the server, and handles changes done by other clients broadcasted by the server.

The frontend Tools typically are associated with one of KES's multiple views of the HK specification. For example, KES's structural view has the main objective of enabling manipulation, inspection, and querying of HK specifications viewed as a graph. These three core features are supported by the `Renderer`, `Inspector`, and `Query` tools, respectively. Meanwhile, the ontology modeling view focuses on enabling the user to visualize ontologies expressed within an HK specification in a more compact form, while also providing shortcuts for the creation of concept taxonomy, instances and links in multiple contexts. These features are mainly supported by five tools. (1) `Context Hierarchy` component enables editing and navigating the context hierarchy. (2) `Concept Hierarchy` component allows visualizing the concept taxonomy specified within a context. This component supports adding concept children and siblings using a predicate chosen by the user among the configured ones. Predicates are represented as HK connectors. The user might, for example, choose to consider a specific connector such as "subclass of" or "subconcept of" to generate the taxonomy (*i.e.*, the connector used to identify the elements that form the hierarchy) visualization at one moment and then choose to visualize the taxonomy based on another connector such as "subproperty of" to visualize the property taxonomy instead. This tool also allows selecting multiple connectors at the same time for computing the taxonomy tree. (3) `Concept Instances` component enables creating and visualizing instances of a concept, also using configurable connectors, similarly to the Concept Hierarchy tool. (4) `Concept Links` component enables visualizing and editing the links of the currently selected entity. Links visualization can be grouped by the entity role in the link or by the connector class (*e.g.*, hierarchy, fact, causal, reasoning, etc.). (5) `Concept Properties` component supports the editing of the HK properties of an entity.

In future versions of this system, we intend to support other HK specification views, such as temporal, spatial, and textual views, which implementation will benefit from this multi-view architecture. As for technologies used in the implementation, we used NodeJS [10] combined with HTML5 and CSS3. We have also developed a custom mechanism to modularize the HTML files related to each tool and use SASS [11] to modularize the CSS styles.

[10] https://nodejs.org/en/
[11] https://sass-lang.com/

3 Demo Script

This demo's main goal is to show how KES supports the contextualized modeling of ontologies while also enabling the integration of symbolic concepts represented through the ontologies with non-symbolic content. For this purpose, we defined a dataset containing traffic simulation images extracted from a visual perception benchmark [5] and an ontology that includes concepts defined in different Smart City contexts (*e.g.*, Smart Government, Smart Mobility, Smart Economy). The objective is to show how KES can support the explicit modeling of ontologies over multiple contexts and their integration with non-symbolic content in the Smart City domain. The demo shows the systems's collaborative capabilities by illustrating different users having a synchronized view of the ontology, while also being able to explore it individually by navigating on the taxonomies of concepts and properties. The images will be exploited to demonstrate KES's capability to enrich the ontology with non-symbolic content by linking image fragments with symbolic concepts. The use of an ontology is to illustrate the support of contextualized ontology modeling. Finally, queries using KES will demonstrate its capability to answer contextual queries over the modeled ontology and associated non-symbolic content.

Demo video 1: Collaborative modeling of an ontology with multiple contexts.
`https://ibm.box.com/v/iswc2020-kes-demo2-video1`

Demo video 2: Enrichment of the ontology with non-symbolic content using the structural view combined with the Concept Hierarchy tool.
`https://ibm.box.com/v/iswc2020-kes-demo2-video2`

Demo video 3: Adding instances of related concepts in multiple contexts and querying the concepts using contexts to reduce the query scope.
`https://ibm.box.com/v/iswc2020-kes-demo2-video3`

References

1. Joseph, M., Serafini, L.: Simple reasoning for contextualized rdf knowledge. In: WoMO (2011)
2. Moreno, M.F., Santos, R., Santos, W., Brandao, R., Carrion, P., Cerqueira, R.: Handling hyperknowledge representations through an interactive visual approach. In: IEEE International Conference on Information Reuse and Integration (IRI) (2018)
3. Moreno, M.F., Santos, R.C., dos Santos, W.H., Cerqueira, R.: Kes: The knowledge explorer system. In: International Semantic Web Conference (P&D/Industry/BlueSky) (2018)
4. Moreno, M.F., Santos, R.C.M., Santos, W.H.S.d., Fiorini, S.R., Silva, R.M.d.G.: Multimedia search and temporal reasoning. arXiv preprint arXiv:1911.08225 (2019)
5. Richter, S.R., Hayder, Z., Koltun, V.: Playing for benchmarks. In: IEEE International Conference on Computer Vision, ICCV 2017. pp. 2232–2241 (2017)
6. Serafini, L., Homola, M.: Contextualized knowledge repositories for the semantic web. Journal of Web Semantics 12 (2012)
7. Smeulders, A.W., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Transactions on pattern analysis and machine intelligence 22(12) (2000)