Demo: Tools for Information Fragmentation in Knowledge Graphs*

Sandro Rama Fiorini[1], Guilherme Ferreira Lima[1], and Marcio F. Moreno[1]

IBM Research Brazil
{srfiorini,guilherme.lima}@bm.com, mmoreno@br.ibm.com

Abstract. Integration of symbolic representations with multimodal data is an important problem in multiple domains. The Hyperknowledge Framework (HKF) is a multimodal knowledge representation framework that allows users to integrate non-graph data to knowledge graphs. In this particular demo, we will show how HKF API can be used to more easily integrate and consume raw data fragments in knowledge graphs.

1 Introduction

It is usual for knowledge graphs to encode knowledge that have counterparts in other data types, such as text or images. For instance, entities in a knowledge graph encoding knowledge about a patient's diagnosis might have counterparts in image exams and the patient's textual medical report. Access to both types of data, which we will call information artifacts, is usually segregated, commonly implemented in altogether different systems. We believe that having integrated access for multimodal information in knowledge graphs should simplify the way programmers can access, manipulate and annotate raw, multimodal data with knowledge graph structures. Such simplified access can be particularly beneficial in domains that are knowledge and data intensive, such as Geology and Medicine, where it is necessary to keep track of scientific interpretation from raw data evidence to symbolic interpretations. In Data Science problems, such as in machine learning engineering, tight integration between knowledge graphs and raw data management should facilitate retrieval and creation of new learning tasks.

The Hyperknowledge Framework (HKF) is a multimodal knowledge graph platform, where multimodal data and symbolic information coexist under the same representation. Previous work demonstrate some of its capabilities in dealing with representation multimodal information in knowledge graphs [3].

In this demo, we demonstrate a new HKF API and a representation scheme to handle reference, access and usage of information artifacts. More specifically, it is an implementation of the General Fragment Model (GFM) [1] for handling representation and resolution of media fragments. GFM introduces a scheme to construct resolvable references to fragments of virtually any type of raw data. It improves on existing frameworks for reference specification, such as W3C Media Fragments 1.0 [4], by providing a

model that is independent of an specific media type. It can also complement document and information description models, such as [2], with a base format for constructing information artifact identifiers based on the structure of such artifacts. HKF API implements GFM, providing a material syntax for programmers to create references to and dynamically extract parts of information artifacts, allowing these to be linked to other entities in the graph. The main contribution of HKF API is that the very descriptions of fragmentation operations serve as identifiers for the fragments themselves within the knowledge graph. So, for example, the operation extracting a rectangular blob from an image becomes the identifier of the blob, which can then be linked in the knowledge graph or retrieved for further processing. In this demo, we demonstrate some of these operations are handled in our framework with a small example.

2 General Fragment Model

The *General Fragment Model* defines a formal model for information reference. It describes a conceptual structure that can be instantiated to create resolvable reference names for parts of information artifacts. An *information artifact* is a codification of some propositional content that realized by some physical or virtual object. Examples are images, text, drawings, sound files, sensor readings, databases and ontologies.

GFM establishes that fragments of information artifacts are specified by *anchors*. An anchor on an information object $o \in O$ is defined by an *indexer* function

$$f(o, d) : O \times D \to O'$$

that maps an arbitrary *token* $d \in D$ to a set of parts O' of O. The tokens in D can be any other information artifacts, especially vectors, dictionaries or strings. For instance, given a text document e, the text fragment e' between characters 10 and 20 can denoted by the application of an indexer function $subtext$ to the target e and with the argument token $[10, 20]$. In this case, the function application $subtext(e, [10, 20])$ is said to be an *anchor* on e and it is a *reference* (or a name) to e'.

Anchors can be composed by other fragments. For example, consider an indexer function $rect$ that extract sub-images from figures and an indexer $channel$ that takes an specific color channel of an image. Considering an image img, we can define the fragment

$$channel(rect(img, [10, 10, 20, 20]), \text{"}\texttt{blue}\text{"})$$

which takes the blue channel of a 20-pixels square positioned at coordinates 10×10 on img. We can go even further by composing the same indexer function multiple times:

$$xywh(channel(xywh(img, [10, 10, 20, 20]), \text{"}\texttt{blue}\text{"}), [2, 2, 4, 4])$$

which denotes a 4-pixel square fragment of the blue channel. Specific implementations might decide on the applicability of a given indexer function on a specific type of information artifact.

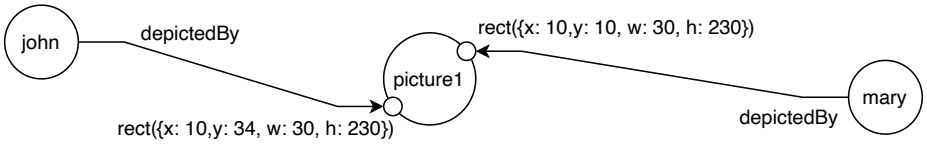

Fig. 1: Simplified 3-node knowledge graph depicting how FI anchors are used to define anchors in HKB.

3 Hyperknowledge Base

The HypeKnowledge Base (HKB), is a knowledge graph database part of HKF [3]. It has been used as a knowledge base in applications for domains ranging from Sports to Agriculture, with particular success in Oil & Gas. It is based on a hypergraph model, where n-ary links associate multiple nodes. It is also a property graph, where links and nodes have their own properties. Sub-graphs can be compartmentalized into *contexts* in which nodes can be imported. More importantly for the discussion in this abstract, it allows representation of raw data as part of the knowledge graph itself, in specific nodes called *content nodes*. These nodes work as any other node in the graph, but can be resolved into the media they represent, including images, text, videos and 3D models.

All nodes might have associated *anchors*. Anchors represent a fragment of the inhering node. In our new model, an anchor is identified by what we call Fragment Identifiers, which allows representation of GFM constructs. In the following, we briefly describe (a) the FI language and (b) the anchor resolver API for this service.

Fragment identifiers (FI) is concrete language for entity naming that implements GFM. It allows definition of fragmentation operations based on a JavaScript-like syntax. The basic form is `artifact.indexer(token)`. Artifacts can be any node identifier in HK. Indexers are usually function names. Tokens can be lists or json-like objects. For instance, given a content node identified by `document`, our previous example of $subtext$ anchor can be specified as:

$$\texttt{document.subtext}(\{\texttt{start} : 10, \texttt{end} : 20\})$$

FIs allow for anchor composition as well. Considering an information artifact `figure` representing an image, the following anchor composition is a valid FI:

$$\texttt{picture1.rect}(\{\texttt{x} : 10, \texttt{y} : 10, \texttt{w} : 20, \texttt{h} : 20\}).\texttt{channel}(\{\texttt{c} : \texttt{"blue"}\})$$

HKB currently allows users to specify and resolve FI anchors on content nodes. In practice, these features allow HKB data processing capabilities within the graph database itself. As mentioned before, content nodes are nodes carrying some raw data. Their fragments are represented as FI anchor strings associated to them. Fig. 1 depicts a simple 3-node graph where two KG nodes representing John and Mary are related to their respective depictions in fragments of `picture1` specified as FI fragments.

The semantics of these anchors is given by a FI resolution API (Fig. 2). The base API is a REST endpoint implemented within the knowledge graph engine. Currently, we have API bindings for JavaScript and Python, the latter of which we show in this demo.

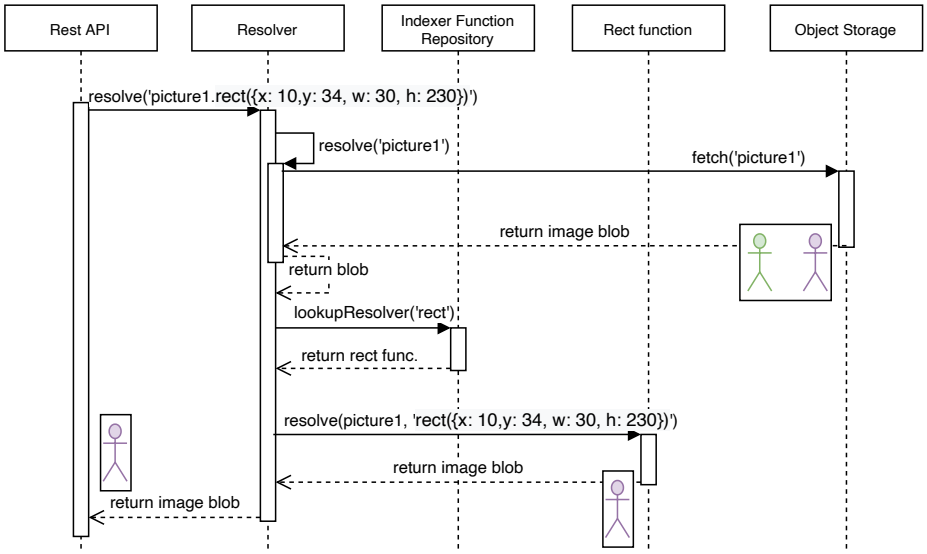

Fig. 2: Sequence diagram of FI resolution on one of the anchors in Fig. 1. The Resolver component starts resolving the FI by recursively resolving the artifact component of the FI. In this case, the artifact is a direct reference to an image *picture1*, which is fetched from an object storage in HKB. The indexer function implementation for *rect* is then retrieved from a repository and resolved on the picture blob with the fragment FI specification. The resulting image image fragment is returned to the Resolver and outputted.

The FI resolution endpoint is able to take full FIs (i.e., artifact and multiple anchors) and recursively resolve them to produce the actual fragment of the content nodes in the knowledge graph. Each indexer function is coded as simple pluggable modules[1] using a standard API provided by HKB. Fig. 2 shows the resolution of one of the anchors in Fig. 1.

4 The Demo

In our demonstration, we plan to demonstrate creation and access to media fragments in a simple example using our Python API and a Jupyter notebook:

1. Demonstration of basic Hyperknowledge constructs in KES (HKF's KG UI);
2. Ingestion of text and image files as content nodes in a knowledge graph via the Python API;
3. Creation and resolution of simple and composed fragments on the ingested files;
4. Demonstration of the creation of these operations in KES;
5. Association of the created fragments with a simple domain ontology.

[1] These modules can be implemented based on existing information and metadata extraction toolkits, such as Apache Tika (https://tika.apache.org) and other components of Apache UIMA (http://uima.apache.org).

References

1. Fiorini, S.R., dos Santos, W.S., Mesquita, R.C., Lima, G.F., Moreno, M.F.: General fragment model for information artifacts (2019), arXiv:1909.04117
2. Groth, P., Gibson, A., Velterop, J.: The anatomy of a nanopublication. Information Services & Use **30**(1-2), 51–56 (2010)
3. Moreno, M.F., Brandao, R., Cerqueira, R.: Extending hypermedia conceptual models to support hyperknowledge specifications. Int. J. Semant. Comput. **11**(01), 43–64 (mar 2017)
4. Troncy, R., Mannens, E., Pfeiffer, S., Van Deursen, D.: Media fragments uri 1.0 (basic). W3c recommendation, W3C (2012)