Knowledge Graph Alignment as a Service

Rob Upson[1,*], Ernesto Jiménez-Ruiz[1,2,*]

[1]City, University of London, London
[2]SIRIUS, University of Oslo, Norway

Abstract

Researchers and practitioners in the field of knowledge graphs rely on a variety of tools to perform knowledge graph alignment. In order to broaden the domain of a knowledge scientist, the process of aligning knowledge graphs must be accessible to a broader range of users, and in a way that can scale to meet the needs of a growing userbase. This paper presents a cloud-based, language agnostic approach that aims at providing knowledge graph alignment tooling at scale.

Keywords

Knowledge Graph Alignment, Cloud Computing, REST API,

1. Introduction

Knowledge graph (KG) alignment[1] is key to integrate knowledge graphs with overlapping domains [1]. Despite the impressive state-of-the-art in KG alignment, only a very few systems provide an easy to use interface. This poses a significant barrier to widespread consumption of these systems, as its installation and subsequent usage requires a level of technical expertise that may not be possessed by research communities beyond computer science. It benefits the industry as a whole, as well as individual researchers, if KG alignment systems can present their functionality in such a way that is abstracted from their implementation, providing the alignment service without the complexity of creating an environment in which to perform it. In order to achieve this, the tools and programs used in the alignment process must be made portable and reproducible in a predictable, unopinionated way to encourage mass adoption.

`LogMap` [2, 3] is a well-known knowledge graph alignment system in the community and part of its success relied on the simplicity of performing alignment via its Web interface.[2] However, the interface does not provide an API to communicate with other systems programmatically, which limits the adoption of `LogMap` in more complex scenarios. The Matching EvaLuation Toolkit (MELT) [4] facilitates the evaluation of KG alignment systems by providing a common interface for the alignment process. However, the MELT platform still has dependencies on the Java runtime and user's familiarity with the command line. In order to use a tool like `LogMap`

International Semantic Web Conference (ISWC) 2022: Posters, Demos, and Industry Tracks, October 23–27, 2022
*Corresponding author.
robert.upson@city.ac.uk (R. Upson); ernesto.jimenez-ruiz@city.ac.uk (E. Jiménez-Ruiz)

[1]The term knowledge graph alignment supersedes the notion of ontology alignment or matching to emphasise the need of aligning not only the terminological part (*i.e.*, concepts and relationships), but also the data (*i.e.*, instances of the concepts).
[2]http://krrwebtools.cs.ox.ac.uk/logmap/

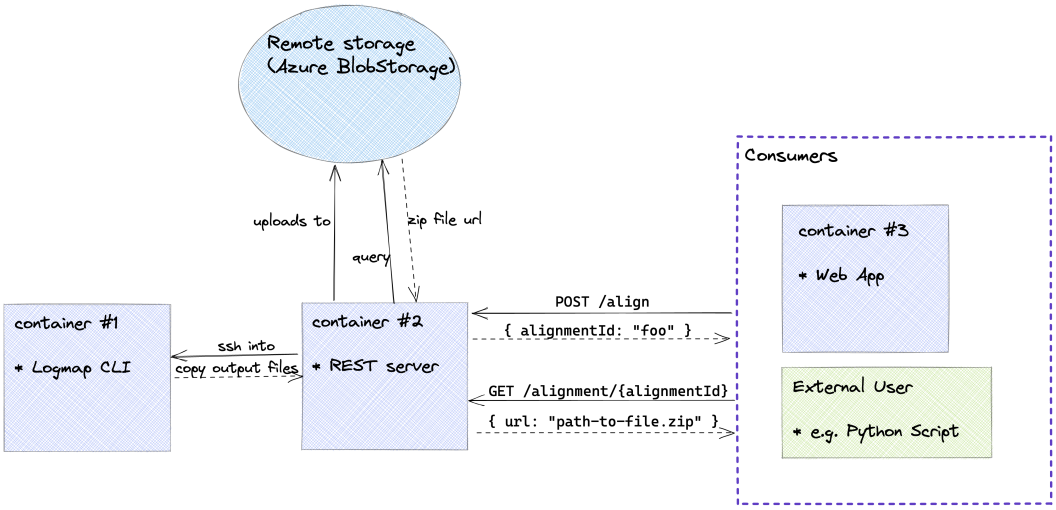

Figure 1: High level overview of container configuration and intercommunication

with the MELT platform, one must first use Maven to produce Docker images, then provide the images to the MELT client. This is useful because any matching system could be swapped into the MELT client as a Docker image, provided it meets the contract, but it does not solve the problem of a zero-dependency alignment system.

This paper presents a proof of concept system that provides the matching functionality of LogMap over a HTTP RESTful API. This gives the consumer absolute control and agency over how to integrate with LogMap in a way that best fits their use case - decoupling LogMap from any programming language or paradigm. A Web app has also been produced that consumes the new API, meaning that a user can use a UI on the Web to perform alignments if this is preferable to a programmatic approach.

2. Architecture Overview

The proposed Knowledge Graph Alignment as a Service (KGAS) system is comprised of 3 individual containerised applications (see Figure 1), deployed onto cloud infrastructure using Azure. Azure is the cloud provider chosen to host the working proof of concept on the Web, but the system is designed such that it could be deployed anywhere, be that to another cloud provider or run on a local server. It was important not to change LogMap's source code directly for the project. The purpose of the proof of concept is to show that the functionality of alignment tools can be exposed without impacting the implementation. This way LogMap can still be developed as a standalone tool, and KGAS will continue to work alongside it without needing to make changes. This was achieved by creating a Docker image that clones LogMap's source code from Git using the latest commit hash, and builds the application from source. This means that if a new release of LogMap were to be made, the app could simply rebuild with the new commit hash and users get the latest version of LogMap immediately. Once there is a dockerised version of LogMap running in a container, the REST API needs to be able to use that app. Since the API

runs in a different container to LogMap, there needs to be an interface for them to communicate. The LogMap container exposes a port accepting SSH connections for this, which enables the CLI commands to be run by the REST server via SSH.

2.1. Cloud Implementation

The containerised applications are platform-agnostic, however, in order to serve the KG alignment outputs as zip files, some external storage is required, which is where the proof of concept becomes coupled to Azure. When a user starts an alignment via the API, a unique ID is generated for the process. This is the ID a user will query against to retrieve the results of the alignment, and also represents the file name used when writing to the remote storage.

For the project to be a success, the solution must be repeatable. To ensure this, all of the cloud resources were created using Infrastructure as Code (IaC); specifically, Terraform.[3] This means that with access to the codebase anyone could spin up their own version of KGAS on their own infrastructure. This provides a wide range of benefits to maintainability of the project as well as scalability.

3. Web Interface

The KGAS Web app created for the proof of concept is a small React application[4] with a UI allowing a user to upload two KGs for alignment, begin the alignment, and query for results of ongoing or previously completed alignments by their id (see Figure 2). The UI will store the alignment ID in the browser's storage, so there is no requirement for a user to remember them. The Web app's purpose is twofold. Firstly, to present a zero-dependency, out of the box way to perform KG alignment. Secondly, it serves as an exemplar piece on how to integrate with the REST API to access LogMap's functionality and retrieve alignment results. This will be helpful for those wishing to integrate with the API programmatically, for example when running an alignment in a Python script.

4. Impact

The working concept shows that KG alignment can be decoupled from its implementation and provided to consumers. This iteration is limited to LogMap specifically, but the same process of containerising and wrapping in a REST API layer can be applied to any alignment system. LogMap itself has more functionality than that which is exposed by KGAS, including the repair of a given alignment or the division of the KG alignment process into subtasks. Exposing these additional behaviours only requires a more mature API, where various parameters or configurations can be passed as arguments. The most significant room for expansion is to decouple from LogMap altogether. Currently KGAS relies on ssh-ing into a machine with the LogMap CLI, then running LogMap specific commands inside that machine. If the interface through which alignments were performed could be generalised, then KGAS would not need to know specific commands for any

[3] https://www.terraform.io/
[4] https://reactjs.org/

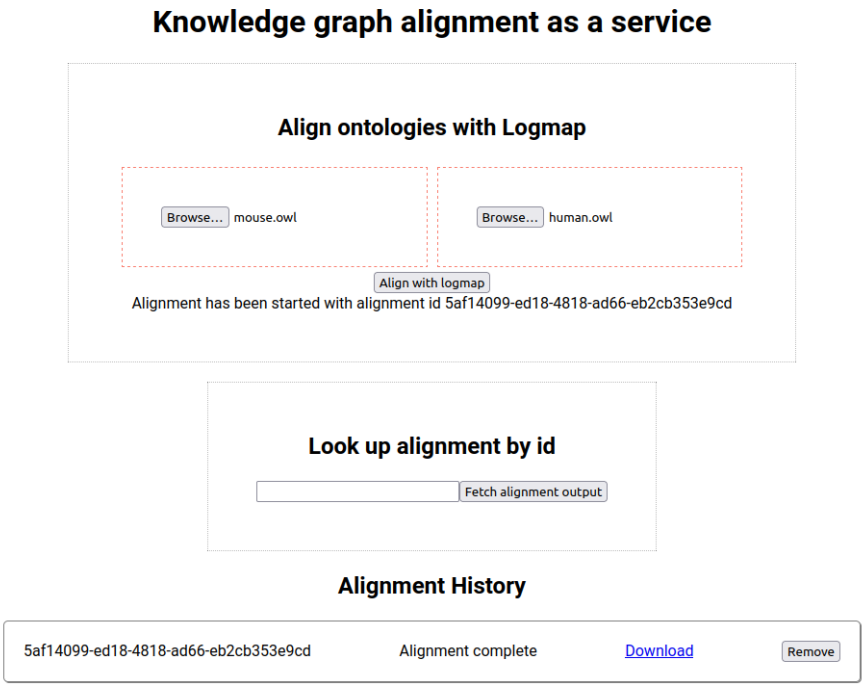

Figure 2: The Web interface

of the systems. The MELT platform provides exactly this. MELT relies on being provided a Docker image for an alignment tool, which it will use for alignment using its common interface. A container could be created ro run MELT and include pre-made images saved to the disk that are compatible with MELT (many of these already exist). Then the API layer only needs to know the CLI commands to interact with MELT, and it can use any one of the alignment tools baked into the image. This will enable a comprehensive KG alignment API where a user can pick and choose the best tool for their use case on demand without having to learn or use any additional technologies. It will also provide a platform where the various tools can be compared with one another, since they will all be in one place and running on the same hardware. The working concept in this demo shows that such a system is within reach.

5. Demonstration

For the demonstration, we will upload to the Web app KGs from the Ontology Alignment Evaluation Initiative[5] like the `human.owl` and `mouse.owl` anatomy ontologies.

The Web app will begin the alignment and display the id. It will then begin polling for results until the alignment completes, at which point the UI will present a button to download the alignment output as a zip file, which can be extracted to view the individual files. In the case of

[5]http://oaei.ontologymatching.org/

LogMap, the alignment output is composed by the following files:

- logmap2_mappings in table format (.txt and .tsv), OWL format (.owl), and RDF alignment[6] format (.rdf): final output mappings selected by LogMap.
- logmap_anchors (same formats as above): mappings computed by LogMap with very high confidence.
- logmap_discarded_mappings.txt and logmap_hard_discarded_mappings.txt: set of mapping that were considered by LogMap but eventually discarded.
- logmap_logically_conflicting_mappings.txt: mappings that were discarded during the logic-based repair process [3].
- logmap_overestimation.txt: set of all mappings considered by LogMap, which represents a manageable subset of the Cartesian product between the entities of the input KGs.
- module1_overlapping_logmap2.owl and module2_overlapping_logmap2.owl: represent the locality modules of the input KGs taking as signature the entities within LogMap's output mappings (see context of an alignment in [5]).

The demo will be performed within a Web browser, and no additional dependencies will be required. The KGAS source code, API documentation and links to the Web app and REST API are available on GitHub: https://github.com/rupson/knowledge-graph-alignment-as-a-service.

References

[1] J. Euzenat, P. Shvaiko, Ontology matching, Springer, 2007.

[2] E. Jiménez-Ruiz, B. Cuenca Grau, LogMap: Logic-Based and Scalable Ontology Matching, in: 10th International Semantic Web Conference, 2011, pp. 273–288.

[3] E. Jiménez-Ruiz, B. Cuenca Grau, Y. Zhou, I. Horrocks, Large-scale interactive ontology matching: Algorithms and implementation, in: 20th European Conference on Artificial Intelligence (ECAI), 2012, pp. 444–449. doi:10.3233/978-1-61499-098-7-444.

[4] S. Hertling, J. Portisch, H. Paulheim, MELT - Matching EvaLuation Toolkit, in: 15th International SEMANTiCS Conference, 2019, pp. 231–245.

[5] E. Jiménez-Ruiz, A. Agibetov, J. Chen, M. Samwald, V. Cross, Dividing the ontology alignment task with semantic embeddings and logic-based modules, in: 24th European Conference on Artificial Intelligence (ECAI), 2020, pp. 784–791. doi:10.3233/FAIA200167.

[6] https://moex.gitlabpages.inria.fr/alignapi/format.html