KnowGL: Knowledge Generation and Linking from Text

Gaetano Rossiello*, Md. Faisal Mahbub Chowdhury*, Nandana Mihindukulasooriya, Owen Corneel, Alfio Massimiliano Gliozzo
IBM Research AI
Thomas J. Watson Research Center, NY

Abstract
We propose KnowGL, a tool that allows converting text into structured relational data represented as a set of ABox assertions compliant with the TBox of a given Knowledge Graph (KG), such as Wikidata. We address this problem as a sequence generation task by leveraging pre-trained sequence-to-sequence language models, e.g. BART. Given a sentence, we fine-tune such models to detect pairs of entity mentions and jointly generate a set of facts consisting of the full set of semantic annotations for a KG, such as entity labels, entity types, and their relationships. To showcase the capabilities of our tool, we build a web application consisting of a set of UI widgets that help users to navigate through the semantic data extracted from a given input text. We make the KnowGL model available at https://huggingface.co/ibm/knowgl-large.

Introduction and Related Work
A Knowledge Graph (KG) is defined as a semantic network where entities, such as objects, events or concepts, are connected between them through relationships or properties. KGs are organized in multi-graph data structures and stored as a set of triples (or facts), i.e. (Subject, Relation, Object), grounded with a given well-defined ontology (Hogan et al. 2021). The usage of formal languages to represent KGs enables unambiguous access to data and facilitates automatic reasoning capabilities that enhance downstream applications, such as analytics, knowledge discovery or recommendations (Mihindukulasooriya et al. 2022).

However, building and curating KGs, such as Wikidata (Vrandečić and Krötzsch 2014), requires a considerable human effort. Systems such as, NELL (Carlson et al. 2010), DeepDive (Niu et al. 2012), Knowledge Vault (Dong et al. 2014), DiffBot (de Sá Mesquita et al. 2019) implement Information Extraction (IE) methods for automatic knowledge base population. A standard IE pipeline consists of several steps, such as co-reference resolution (Dobrovolskii 2021), named entity recognition (Wang et al. 2021), relation extraction (Zhong and Chen 2021), and entity linking (Wu et al. 2020), each of which is commonly addressed as a separate task. A pipeline approach presents several limitations, e.g. error propagation among different IE components and complex deployment procedures. Moreover, each component of the pipeline is trained independently using different architectures and training sets.

The ability of generating structured data from text makes sequence-to-sequence Pre-trained Language Models (PLMs), such as BART (Lewis et al. 2020) or T5 (Raffel et al. 2020).
et al. 2020), a valuable alternative to successfully address IE (Glass et al. 2021, 2022; Ni et al. 2022; Wu, Zhang, and Li 2022; Cabot and Navigli 2021; Josifoski et al. 2022), entity/relation linking (Cao et al. 2021; Rossiello et al. 2021) and semantic parsing (Zhou et al. 2021; Dognin et al. 2021) tasks. In this work, we further explore this direction by asking whether PLMs can be fine-tuned to read a sentence and generate the corresponding full set of semantic annotations that are compliant with the terminology of a KG. For this purpose, we propose a framework able to convert text into a set of Wikidata statements. As shown in (Mihindukulasooriya et al. 2022), KnowGL parser can be used to automatically extract KGs from collections of documents, with the purpose to help users with semantic content exploration, to create trend analysis, to extract entity infoboxes from text, or to enhance content-based recommendation systems with semantic features.

KnowGL Parser

Figure 1 shows an overview of the Knowledge Generation and Linking (KnowGL) tool. Given a sentence as input, KnowGL returns a list of triples (subject, relation, object) in a JSON format. From the example, KnowGL identifies the entity mentions semantic web and inference rules in the sentence and generates the relation label use between them. For each mention, the output provides the corresponding entity label and its type. If the entity labels, entity types and relation labels are found in Wikidata, then KnowGL also provides the Wikidata IDs associated with them. As shown in Figure 1, KnowGL Parser consists of three main components: generation, ranking and linking, as described below.

Knowledge Generation We address the fact extraction as an autoregressive generation problem. In other words, given a natural language text input, the knowledge generation model generates a linearized sequence representation that contains a set of facts expressed in the textual input. We adopt the following schema to represent the semantic annotations of a triple in the target sequence: [(subject mention # subject label # subject type) | relation label | (object mention # object label # object type)]. If the input text contains multiple mention pairs, the linearized target representations are concatenated using $|$ as a separator, and the facts are sorted by the order of the appearances of the head entity in the input text. Unlike in (Cabot and Navigli 2021; Josifoski et al. 2022), we generate the surface forms, entity labels, and type information for both head and tail entities in the target representation. This represents a full set of semantic annotations, i.e. ABox and TBox, to construct and populate a KG with new facts. Our hypothesis is such self-contained fact representation also acts as an implicit constraint during decoding. We exploit BART-large (Lewis et al. 2020) as the base model and cast this as a translation task where at training time the encoder receives a sentence, and the decoder generates the sequence target representation as described above. To train the generation model, we extend the REBEL dataset (Cabot and Navigli 2021) by adding the entity labels and their types from Wikidata for each entity surface form in the text. REBEL is an updated and cleaner version of T-REx (ElSahar et al. 2018), a distantly supervised dataset for relation extraction built by aligning Wikipedia abstracts with Wikidata triples. We use cross-entropy loss as standard in machine translation whereas, in teacher forcing, the model regards the translation problem as a one-to-one mapping process and maximizes the log-likelihood of generating the linearized facts given the input text. As reported in (Mihindukulasooriya et al. 2022), our KnowGL model outperforms (F1 = 70.74) both a standard IE pipeline system (F1 = 42.50) and the current state-of-the-art generative IE model (F1 = 68.93) (Josifoski et al. 2022). For the evaluation, we use the test set released with the REBEL dataset.

Fact Ranking This component parses the target sequences generated by the knowledge generation model using a regular expression. The goal is to create a ranked list of distinct facts with their scores. We extract facts from all the returned sequences generated by each beam (where the number of beams is a hyper-parameter). For each extracted fact we consider the negative log-likelihood of the entire generated sequence as a score. Since the same fact can appear in different returned sequences, we sum the scores of each sequence where the fact occurs. The idea is to promote those facts/triples that occur multiple times in different beams. Finally, the facts are sorted by their scores.

Linking to Wikidata The linking component enables retrieving the Wikidata IDs associated with the generated entity, type and relation labels. For efficiency, we create label-to-IDs maps from Wikidata and store them in key-value data storage systems to avoid bottlenecks caused by running multiple SPARQL queries to a Wikidata triple store. It is worth noticing that the model can generate new entity, type, or relation labels that are not in Wikidata. In this case, the linking component returns a null ID and the triple can be used as a candidate for adding new facts in Wikidata.

Demonstration

KnowGL Parser is implemented in Python language and deployed as a REST API using the Flask framework. The input is a sentence and the output is a JSON format structured as shown in Figure 1. The user interface described in our video demonstration is implemented as a separate web application using Node.js and React web tool frameworks. The UI allows users to insert textual content using a textbox. Then, the returned JSON is parsed by the UI enabling different types of visualizations. For instance, the facts can be organized in a directed multi-graph where the nodes are the entities and the edges represent the relations between two entities. The user can navigate and interact with the nodes and edges to easily locate the textual evidence associated with the triples.
