WikidataComplete: Knowledge Graph Completion using Question-Answering

Kunpeng Guo[1,3], Bernhard Kratzwald[2], Dennis Diefenbach[1,3], and Christophe Gravier[1]

[1] CNRS, Laboratoire Hubert Curien UMR 5516, Université Jean Monnet
[2] Chair of Management Information Systems, ETH Zurich
[3] The QA Company, Saint-Etienne, France

Abstract. In this demonstration, we showcase `WikidataComplete`, a KG completion system for Wikidata based on Question Answering (QA). We showcase (https://wikidatacomplete.org) that it is possible to leverage Question Answering to extract new facts from Wikipedia and jointly provide explanations for these extracted facts. Providing explanations has two advantage: first it helps human annotators to verify if the generated facts are correct, second it allows ingesting the approved facts in the KG together with provenance information. Ultimately, we measure the accuracy of such a system for the first time on a real scenario including human annotations.

Keywords: Knowledge Graph Completion, Question Answering, Natural Language Processing, Human in the loop.

1 Introduction

This demo paper discusses Question Answering (QA) based Knowledge Graph (KG) completion. Knowledge graphs (also known as knowledge bases) are an important data repository for a number of Natural Language Processing (NLP) tasks like entity linking [6], query expansion [1], and QA [3]. While KGs are critical for the performance of these downstream NLP tasks, they are however often incomplete. For example between the Wikidata entities of type *TV program* (`Q15416`), 96% do not have a *creator* and 94% do not have a *production company* (`P272`). KG completion can be loosely grouped into two paradigms. The first method is KG completion via link prediction which predicts missing relation based on the existing graph structure [5, 7]). The second method is to complete the KG by leveraging missing relations from free text [8, 9]. `WikidataComplete` falls into the second category. For the sake of the demonstration, we will showcase to the audience the facts that `WikidataComplete` can leverage for a finite set of 20 Wikidata properties taken from different Wikipedia categories (TV programs, Sport, Pet, Magazine, Company, Video Games…). We give access to the 6,045 newly extracted facts as well as to the lightweight annotation interface that we offer to editors. On the demo platform, the

[1] Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

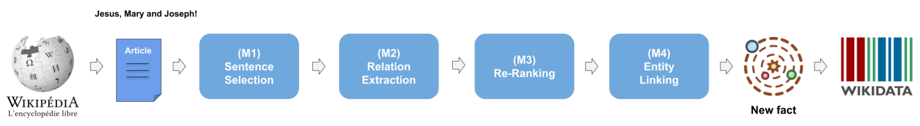

Fig. 1: QA-based fact extraction pipeline in `WikidataComplete`.

conference attendee will see facts that are not yet approved by an editor. The attendee will see the explanations for the extracted facts and be able either to approve or reject them. If the user is logged in Wikidata, the new assertion will be added to Wikidata on the spot at the conference booth. `WikidataComplete` builds upon [4] for the fact extraction system. We recap that extraction process in Section 2 and we provide a new real-field evaluation of the system based on human annotators in Section 3.

2 QA-based fact extraction

We build upon [4] for extracting and selecting new facts from Wikipedia using QA. We recap this approach using a running example based on the Wikidata entity `Q11066637` (the television program "Jesus, Mary and Joseph!"). For the sake of illustration, we assume that we would like to extract the missing information about the property "original broadcaster" (`P449`) – the possible properties for a given entity are assumed to be known from the KG ontology. We will later discuss how this generalize to other entities and properties. `WikidataComplete` starts with the Wikipedia page corresponding to the entity `Q11066637`. Then, a four step process (M1–M4) extracts information about the missing property(see Figure 1).
(M1) Sentence Selection takes the Wikipedia article of `Q11066637` together with the possible labels ("original broadcaster", "original channel", "channel", "TV channel") for the property `P449` . The module splits the whole article into sentences and then ranks them by the cosine-similarity scores calculated on their Tf-Idf vectors.
(M2) Relation Extraction receives the ranked sentences and extracts a text span that is appropriate to fulfill the missing property. This task is accomplished by casting the problem into a QA task. In the illustrative task, the question should be "Who is the original broadcaster of 'Jesus, Mary, and Joseph!'?", but the generated question is simplified to "original broadcaster?" to avoid the human supervision in generating questions. BERT [2] is used to encode the inference, but the final softmax layer over different answer spans is removed with the goal of aggregating different results from the sentences. This process is repeated for all ranked candidate sentences given by M1.
(M3) Re-Ranking receives all the candidate answers in text span and generates statistical features (span score, answer length, etc.) that maps the answer with the question. The module re-ranks all the candidate answers using these features.
(M4) Entity Linking links the text span to the corresponding entity in Wikidata. This is accomplished by using a pre-trained KG linker model (see [4]) which significantly improves the precision of linking and avoid mislinking in ambiguous conditions. In addition, it helps to filter out some irrelevant answers produced by the module M2.

We use the described pipeline (M1–M4) to extract new facts in two different ways. First, we extract new facts based on a fixed relation as illustrated by our running example in this section for `P449`. We also run it for a given category (e.g., facts about television programs, sport, or celebrities). In this case, we first compute which are the most used properties for instances of a specific category. Then we consider all instances having an associated Wikipedia article were these properties are missing. In our demonstration, the user is able to select facts either by selecting a specific category or by selecting a specific property. The fact distribution for the categories presented in the demo is the following: *TV program* (`Q15416`) : 1573 facts, *Sport* (`Q349`) : 746 facts, *Pet* (`Q39201`) : 107 facts, *Magazine* (`Q41298`) : 536 facts, *TV series* (`Q5398426`) : 1759 facts, *Company* (`Q783794`) : 543 facts, and *Video Games* (`Q7889`) : 781 facts.

3 User-Verification of the generated facts

Each newly extracted fact is presented to the user together with the corresponding evidence – the sentence with the fact utterance. Links to the corresponding Wikidata and Wikipedia page are provided so that the user can explore the corresponding entity and textual description. Based on this information a user is able to approve, reject, or skip a newly proposed fact. Note that the editing effort of a user is reduced just to a click on yes, no, or skip. If approved, the fact is also added to Wikidata as a statement. The new statement is inserted together with a reference to the Wikipedia article.

Benchmark vs Human annotation							
	Pb(1)	Pb(2)	Ph		Pb(1)	Pb(2)	Ph
P178 (Developer)	41.5%	62.4%	73.1%	P404 (Game mode)	44%	87.6%	62%
P407 (Lang. of work)	34.8%	73.8%	88.67%	P4743 (Animal breed)	78.8%	97.3%	68%
P495 (Country origin)	59.9%	83.7%	56.1%	P527 (Has part)	8%	15%	9.6%
P641 (Sport)	53.3%	85.3%	15.38%	P749 (Parent org.)	39.1%	52.9%	28%
P1056 (Product or mat.)	5.3%	9.3%	12%	P123 (Publisher)	36.8%	57.3%	60%
P127 (Owned by)	12.5%	26.7%	30%	P161 (Cast members)	63.6%	75%	51%
P17 (Country)	42.9%	93.3%	56.9%	P19 (Place of birth)	27.8%	40.1%	14%
P364 (Film/TV orig. lang.)	35.8%	75.8%	82%	P400 (Platform)	15.5%	50%	76%
P437 (Distribution format)	2%	6.9%	34%	P449 (Orig. broadcaster)	55.5%	73.4%	86%
P57 (Director)	34.2%	43.1%	42.3%	P921 (Main subject)	5.3%	18.3%	72%

Table 1: Performance comparison for 20 randomly selected properties between distant supervision (Pb1) and human annotators (Ph). (Pb2) is the same as Pb1 where we remove triples for which the object cannot be linked.

The system performance was originally reported to be 49.4%. The performance for a subset of properties is provided in Table 1. Nonetheless, the dataset was built using distant supervision: it is not guaranteed that when the correct answer is found also the corresponding evidence is correct. Along this demo, we newly compare the performance of the benchmark created using distant supervision [4] against human annotations. We make this comparison for the 20 randomly selected Wikidata properties in Table 1. We started by sampling Wikidata triplets that meet the following two requirements:

(i) the Wikipedia article of the subject entity can be found; (ii) in the Wikipedia article of the subject at least one of object entity's labels appears. We choose 1,000 examples for training the system and evaluated it on a different set containing 500 examples. The performance results (Pb) for each property are reported in Table 1. We report the performance of the pipeline over all the statements (Pb1) and also the performance of the pipeline over all the statements for which the object could be linked (Pb2). Moreover we measured the performance using human annotators (Ph). For each property human annotators evaluated 50 of the generated facts. We can only present to humans statements for which the object could be linked. The metric (Ph) needs therefore to be compared with the metric (Pb2). The results of (Pb2) and (Ph) are very similar. We can therefore conclude that: first the system is still mostly extracting facts from sentences that are also an evidence for the facts, and second that our selection strategy allows to identify entities for which the missing statements can be found in the corresponding Wikipedia articles.

The current version of the demo is hosted at: https://wikidatacomplete.org. Only facts that were not yet approved by a user are offered – validated ones were added to Wikidata.

4 Conclusion

In this paper, we presented `WikidataComplete`, a KG Completion system, which extracts new facts from unstructured data, i.e. Wikipedia articles, using QA. The system has strong scalability so that it can potentially generate thousands of new pieces of knowledge, which makes a significant contribution to the completeness of KGs. Moreover, based on the convenience of the system, it also greatly facilitates Wikidata editors and contributors. In the future, we would like to investigate how the extraction process can be carried out on the fly so that users can choose the subject and property of their interests directly. We believe that if we could build a more powerful and complete version of such a system, it will significantly boost the KG completion process.

References

1. Dalton, J., Dietz, L., Allan, J.: Entity query feature expansion using knowledge base links. In: SIGIR. pp. 365–374 (2014)
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018)
3. Diefenbach, D., Maret, P.: Qanswer: A question answering prototype bridging the gap between a considerable part of the lod cloud and end-users. In: WWW (2019)
4. Kratzwald, B., Kunpeng, G., Feuerriegel, S., Diefenbach, D.: Intkb: A verifiable interactive framework for knowledge base completion. In: COLING 2020 (2020)
5. Lao, N., Mitchell, T., Cohen, W.: Random walk inference and learning in a large scale knowledge base. In: EMNLP (2011)
6. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: shedding light on the web of documents. In: Semantics (2011)
7. Riedel, S., Yao, L., McCallum, A., Marlin, B.M.: Relation extraction with matrix factorization and universal schemas. In: NAACL (2013)

8. Weikum, G., Theobald, M.: From information to knowledge: harvesting entities and relationships from web sources. In: SIGMOD-SIGACT-SIGART (2010)
9. West, R., Gabrilovich, E., Murphy, K., Sun, S., Gupta, R., Lin, D.: Knowledge base completion via search-based question answering. In: WWW (2014)