Entity Enabled Relation Linking

Jeff Z. Pan[1,2,3], Mei Zhang[1,2], Kuldeep Singh[4], Frank van Harmelen[5], Jinguang Gu[1], and Zhi Zhang[1]

[1] College of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430065, China,
[2] Department of Computer Science, The University of Aberdeen, UK
[3] Edinburgh Research Centre, Huawei, UK
jeff.z.pan@abdn.ac.uk, zhangmeiontoweb@gmail.com, simon@wust.edu.cn, wustzz@sina.com
[4] Nuance Communications Deutschland GmbH, Germany
kuldeep.singh1@nuance.com
[5] Vrije Universiteit Amsterdam, Netherlands, frank.van.harmelen@vu.nl

Abstract. Relation linking is an important problem for knowledge graph-based Question Answering. Given a natural language question and a knowledge graph, the task is to identify relevant relations from the given knowledge graph. Since existing techniques for entity extraction and linking are more stable compared to relation linking, our idea is to exploit entities extracted from the question to support relation linking. In this paper, we propose a novel approach, based on DBpedia entities, for computing relation candidates. We have empirically evaluated our approach on different standard benchmarks. Our evaluation shows that our approach significantly outperforms existing baseline systems in both recall, precision and runtime.

Keywords: Question Answering, Semantic Web, Semantic Search, Predicate Linking, Knowledge Graph

1 Introduction

Over the past years, the number and size of Knowledge Graphs (KG) [24] in the Semantic Web has increased significantly. Among them, well known ones include DBpedia [1], Yago [32], Freebase [5] and Wikidata [37]. To make such information easily available, many question answering (QA) systems over KGs have been created in the last years [16,12,4]. The research community has addressed the problem of question answering over KGs via two different approaches. Firstly, researchers have developed end-to-end QA systems such as [16,8] that use deep learning and machine learning models to directly predict mapping/linking of entities and relations in the input question to their KG occurrences to extract correct answers. These end to end QA systems are frequently developed for question answering over Freebase due to the availability of large training data in benchmarks (e.g. the SimpleQuestion benchmark [6] for Freebase, containing 100.000 questions). However, for DBpedia the availability of training data is

limited to at most 5000 questions [34]). Therefore, researchers have focused on QA systems based on semantic parsing that heavily rely on semantics associated with natural language understanding of the input question. Semantic parsing based QA (SQA) systems implement a sequence of tasks (often referred to the QA pipeline [26]) to translate natural language questions to their corresponding SPARQL query. These systems over DBpedia implement independent component(s) in the architecture for entity and relation linking [29,26], that is, to link the extracted entities and relations from the input question to their knowledge graph occurrences. While doing so, most QA systems face the following challenges: i) how to deal with the extraction of entity and relation candidates in the question, and ii) how to link the relation and entity candidates to the knowledge graph. The third approach is the collaborative QA systems which promotes reusability of QA components.

In this paper, we address the challenge of relation linking. Recently, to build SQA systems in a collaborative effort, many frameworks such as Qanary [7], OKBQA[6] [14] and Frankenstein [31] are developed that use modular approaches for building QA systems by reusing existing independently released tools. Several independent entity and relation linking tools such as DBpedia Spotlight [17], AGDISTIS [36], SIBKB [30], ReMatch [18] and Tag Me [10] are reused in these frameworks. Following this approach, in this paper, we will develop a new relation linking component, embed it in an existing framework (in our case Frankenstein) and compare its performance against the state of the art. We focus on *relation* linking because independent *entity* linking tools already perform well when they are applied to QA frameworks like Frankenstein, while on the other hand all the existing independent relation linking tools fail miserably both in terms of precision and runtime [31]. This failure of relation linking tools impact the overall performance of the QA frameworks. Recently released studies by Singh et. al. [31,28][7] have concluded that one of the main reasons behind relational linking tool having limited performance is that the existing relation linking tools focus more on identifying relations in the original question, while completely ignoring the context of the entities that co-occur with these relations.

Therefore, in this paper, we propose to make use of entities appearing in questions to support the task of relation linking over the DBpedia knowledge graph. More precisely, properties that are logically connected to the target entities (as domains or ranges) are called the *candidate property list* (or simply *property list*). This property list can then be used to expand the set of relation candidates which can be used for the construction of SPARQL queries in the QA pipeline. Our evaluations later in this paper will show that the use of logically connected property candidates leads to substantial gains in not only recall, precision but also runtime.

For example, given an input question "Which comic characters are painted by Bill Finger?", we can extract the relation phrase `are painted by` and the en-

[6] `http://www.okbqa.org/`

[7] Authors evaluated five independent relation linking tools for DBpedia and other 18 entity linking tools

tity phrase `Bill Finger`. Typically, existing relation linking approaches [30,18] would directly extract the relation phrase in the question, while ignoring the entities (Bill Finger in this case) and expand the relation candidate using a synonym list, such as "painter". These tools then attempt to map the relation candidate to DBpedia relations. However, this mapping leads them to provide `dbo:painter`[8], leading to an empty answer to the resulting SPARQL query. The SPARQL query returns null because `dbo:painter` is not the correct property of the entity `Bill Finger`. Instead, we assume that the entity `Bill Finger` is already linked to its DBpedia mention and acts as one of the inputs besides the natural language question. We construct a property list (see more details in the next sections) by collecting all properties of the DBpedia entity `dbr:Bill_Finger`[9], including properties that have some types of `dbr:Bill_Finger` as domains or ranges. We then further make use of the property list, including ranking of the list and making sure that the range of the chosen property is compatible with `comic characters`. At the end, we get `dbo:creator` as a candidate relation. When we apply this result to the SPARQL query corresponding to the question, we can conclude that `dbo:creator` is the best choice. In this way, we eliminate the requirement for large training data by focusing on the structure of the DBpedia knowledge graph (in terms of entities and their associated properties), and by considering the context of the relation consisting of the entities in the question.

Based on the above idea, we propose and implement a new relation linking framework (Entity Enabled Relation Linking, EERL) for factoid questions using DBpedia. The contributions of this paper can be summarised as follows: Firstly, a novel approach for generating and ranking candidate relations to be used in QA systems. Secondly, an efficient implementation of this approach in the EERL framework, that can be deployed as part of a larger QA pipeline; and thirdly, an in-depth evaluation of our approach using a set of questions from two benchmarking datasets having more than 5000 diverse questions. Our evaluation shows a large improvement over the state of the art in both precision, recall and the runtime.

As most existing works in the literature, we test our approach on DBpedia. However, there is no specific assumption in our work on the structure or schema of the underlying knowledge graph, and our method should be equally applicable and can be extended to any other knowledge graph.

The rest of the paper is organised into the following sections: Section 2 presents our problem statement, Section 3 describes some of the major contributions in relation linking used in question answering, Section 4 presents our approach for the identified problem. Section 5 describes our experimental setting and our evaluation results. Finally Section 6 presents some of the key discussion points considering our approach, and we conclude the paper in Section 7.

[8] dbo is bound to `http://dbpedia.org/ontology/`
[9] dbr is bound to `http://dbpedia.org/resource/`

2 Background

2.1 Knowledge Graph

More formally, we define a knowledge graph [24,23] $\mathcal{G}=\mathcal{T}\cup\mathcal{A}$ consisting of a data sub-graph $\mathcal{A}$ (or ABox) and a schema sub-graph $\mathcal{T}$ (or TBox). Facts in the ABox are represented as triples of the following two forms:

- *Relation assertion* (h,r,t), where h (t) is the head (tail) entity, and r the relation; e.g., (`dbr:Barack_Obama`, `dbp:birthPlace`, `dbr:Hawaii`) is a relation assertion.
- *Type assertion* (e, rdf:type, C), where e is an entity, rdf:type is the instance-of relation from the standard W3C RDF specification and C is a type; e.g., (`dbr:Bill_Finger`, rdf:type, `dbo:Person`) is a type assertion.

A TBox includes type inclusion axioms, such as (`dbo:Person rdfs: subClassOf dbo:Agent`), and relation axioms, such as (`(dbp:birthPlace rdfs:domain dbo:` Person)) and (`dbp:birthPlace rdfs: range dbo:Place`). There can be other kinds of type and relation axioms defined in the W3C standard knowledge graph schema language OWL, which is based on Description Logics. We refer the readers to [2] for more details on Description Logic. In the rest of the paper, we use $\mathrm{E}(\mathcal{A})$ (resp. $\mathrm{R}(\mathcal{T})$) to refer to the set of entities (resp. relations) in $\mathcal{A}$ (resp. $\mathcal{T}$). Note that the set of relations in $\mathcal{A}$ is a subset of the set of relations in $\mathcal{T}$, some of which might not have instances in $\mathcal{A}$.

2.2 Problem Statement

Firstly, let us formalise the problem of relation linking for factoid questions, before proposing our new research problem. Given the schema $\mathcal{T}$ of a knowledge graph $\mathcal{G}=\mathcal{T}\cup\mathcal{A}$ and an input natural language question q, the task of entity linking is to identify a set of relations $R_q\subseteq\mathrm{R}(\mathcal{T})$ for the set of relation phrases in q.

In this paper, we propose a variant of the problem of relation linking for factoid questions, based on entities identified within these questions. Formally, given a knowledge graph $\mathcal{G}=\mathcal{T}\cup\mathcal{A}$, an input natural question q and a set of entities $E_q\subseteq\mathrm{E}(\mathcal{A})$ identified in q, the task of *entity enabled relation linking* is to identify a set of relations $R_q\subseteq\mathrm{R}(\mathcal{T})$ for the set of relation phrases in q based on the entities E_q.

Note that entities in the ABox $\mathcal{A}$ of $\mathcal{G}$ as well as their interconnections are not taken into account in the task of (pure) relation linking, but in entity enabled relation linking, which, in fact, also takes into account implicit connections between the entities in $\mathcal{A}$. This makes entity enabled relation linking a (much) harder problem than (pure) relation linking.

3 Related Work

Given a natural language question and a knowledge graph, the task of relation linking is to identify relevant relations from the given knowledge graph for the

relation phrases from the question. There are a variety of resources and systems for relation linking over DBpedia.

PATTY [20]: PATTY is a large resource for textual patterns that denote binary relations between entities. Its a two column large knowledge source, where one column represents natural language relational patterns, and the another column contains associated DBpedia predicates. However, PATTY cannot be used directly as a component for relation linking in a QA system and needs to be modified based on individual developers' requirements.

BOA [11] : BOA can be used to extract natural language representations of predicates independent of the language, if provided with a Named Entity Recognition service. Like PATTY, BOA also needs to be modified before using directly in a QA system.

SIBKB [30] : SIBKB provides searching mechanisms for linking natural language relations to knowledge graphs. The tool uses PATTY as the underlying knowledge source and proposes a novel approach based on semantic similarities of the words with DBpedia predicates for an independent relation linking tool that accepts question as input and provides DBpedia properties as output.

Rematch [18]: The ReMatch system is an independently reusable tool, for matching natural language relations to KB properties. This tool employs dependency parse characteristics with adjustment rules and then carries out a match against KG properties enhanced with the lexicon Wordnet. However, the run time is relatively slow for each question.

EARL [?]: EARL is the most recent approach for the joint entity and relation linking. This tool treats entity and relations linking as a single step. At first, it aims to identify entities in the question and, following a graph traversal approach, identifies the relation associated with the entities. EARL determines the best semantic connection between all keywords of the question by exploiting the connection density between entity and relation candidates.

The work by Usbeck et al.[36] proposes an entity linking tool AGDISTIS that is most closely related to our approach. AGDISTIS combines the HITS algorithm with label expansion strategies and string similarity measures. Similar to our approach where we rely on linked URIs of entities in a question besides the question as input, AGDISTIS accepts a natural language question (or sentence) and recognised entities as inputs to provide disambiguated entity URIs. However, unlike our approach, it is restricted to entity linking and does not perform relation linking. Furthermore, TBSL QA system [35] uses entities in the query Q to generate templates that are later filled with properties from the graph which is quite similar to our idea of the use of entities in finding correct predicate. TBSL uses external resource BOA to find the correct matching besides string matching to get the correct relations. Our approach goes a step ahead and heavily relies on ontology reasoning to find the correct predicate(s) for the given entity without using any external knowledge resource. This demonstrates the power of exploiting knowledge encoded in the knowledge graph itself. Furthermore, using entities to map the relations in a question is new in QA relation linking but this has been well studied in ontology mapping (alignment). For example, in map-

ping tables to ontologies, some approaches such as [19] create a candidate list of the properties and rank them after linking the entities in the table cells. Furthermore, there are also efforts developing rich schema for question answering in e.g., legal domain [9].

4 Approach

In this section, we describe our approach to use entities in E_q for relation linking.

4.1 Preliminaries and Proposed Hypothesis

To evaluate our novel approach for addressing relation linking problem, we have analysed 100 randomly chosen question answering pairs from the benchmarking datasets of Section 5. While analysing the SPARQL query associated with the input questions, We observe that most of predicates of these queries (i.e. the KG relations for the natural language relations occurring in the input question) are the properties of the entities in the questions. For example, given a question "**Which comic characters are painted by Bill Finger ?**" (a question from the LC-QuAD dataset [34]), the SPARQL query of this question is:

```
"SELECT␣DISTINCT␣?uri␣WHERE{
?uri␣␣http://dbpedia.org/ontology/creator
http://dbpedia.org/resource/Bill_Finger.
?uri␣https://www.w3.org/1999/02/22-rdf-syntaxns#type
http://dbpedia.org/ontology/ComicsCharacter.}␣"
```

In this query, the predicate `dbo:creator` of the associated entity `dbr:Bill_Finger` is one of the property of `dbr:Bill_Finger` in DBpedia. Furthermore, it is often a case that there is no natural language label of a relation in the question. For example, the question "**How many shows does HBO have ?** "(a question from LC-QuAD dataset [34]) contains no natural language relation label. Such questions are called questions with hidden relations [28]. The SPARQL Query of this question is:

```
"SELECT␣DISTINCT␣COUNT(?uri)␣WHERE{
?uri␣http://dbpedia.org/ontology/channel
http://dbpedia.org/resource/HBO␣.
?uri␣https://www.w3.org/1999/02/22-rdf-syntaxns#type
http://dbpedia.org/ontology/TelevisionShow␣.}"
```

Here, the predicate `dbo:channel` is not the property of `dbr:HBO` explicitly, but rather a property from the type `dbo:Broadcaster` of `dbr:HBO`, where `dbo:Broadcaster` is a range of `dbo:channel`.

Hypothesis: Based on this analysis, we propose the following hypothesis: *"The relations in questions are properties of the entities occurring in the question or properties of the types of these entities."* This hypothesis is surprisingly simple, but to the best of our knowledge this simple hypothesis has not yet been exploited in any of the current state of the art approaches for QA relation linking.

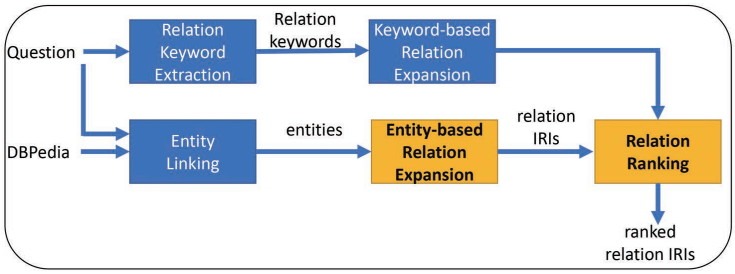

Fig. 1: **Conceptual Architecture of our EERL Framework**. All of the baseline relation linking frameworks found in the literature share the top part of the pipeline. Our contribution in the EERL framework is the additional entity-based part of the pipeline in the bottom half of the image.

Based on this hypothesis, we developed an approach that follows five main steps (see fig. 1): (1) RELATION KEYWORD EXTRACTION: to extract natural language relation keywords from the question, (2) KEYWORD-BASED RELATION EXPANSION: to expand extracted relation keywords using background knowledge, (3)ENTITY LINKING: link the entities in natural language questions to DBpedia IRIs, (4) ENTITY-BASED RELATION EXPANSION: to use entities in E_q to form candidate property list, and then (5) RELATION RANKING: to rank the candidates in property list to get the best relations R_q. We implemented our approach in proposed EERL framework which is described in next section.

4.2 EERL Framework

Our EERL framework consists of five different modules as illustrated in the Fig. 1. The framework has two inputs: a natural language question q and and the DBpedia knowledge graph.

Relation Keyword Extractor The first module is the Relation Keyword Extractor that extracts relation phrases from the input question.

Example: In the question "Which comic characters are painted by Bill Finger?", we extract the "painted by" phrase. We utilize TexRazor API[10] which provides us with relation phrasess and reused the implementation of this module from the work of [30].

Keyword-based Relation Expansion In the second module "Keyword-based Relation Expansion", we expand the relation phrase "painted by" using background knowledge from PATTY [20] to get a list of associated relation phrases. We first convert "painted by" in a vector using Glove [25] and then used the

[10] `https://www.textrazor.com/docs/rest`

vector representation of PATTY created by [30] to get the most suitable relation phrase. For the given example, this step provides us with "painter".

Both of these steps are performed by all or most of the baseline systems, and we do not claim them as novelty. We include them only for completeness of our description and as part of overall offered solution. In a parallel step, input DBpedia IRIs of the entities from the question (dbr:Bill_Finger in this case) are used to create a property list as described below.

Entity-based Relation Expansion This module is the core of our approach, and relies on our proposed hypothesis in the previous section.

Given a KG $\mathcal{G} = \mathcal{T} \cup \mathcal{A}$, the entities in a knowledge graph are the nodes of the $\mathcal{G}$. These nodes are connected to other nodes (i.e. other entities) via directed labeled edges. We divide these edges into two categories: explicit and implicit relations.

Explicit relations. Explicit relations are the properties of entities which can be fetched from $\mathcal{A}$. For example, in the sentence: **The spouse of Barack Obama is Michelle Obama**, represented in RDF as the triple (`dbr:Barack_Obama`, `dbo:spouse`,`dbr:Michelle_Obama`), `dbo:spouse` is the property of `dbr:Barack_Obama`.

Implicit relations. Implicit relations are the relations between entities that can be derived from $\mathcal{T}$. For instance, from the sentence "Barack Obama is born in Honolulu", the explicit relation is `dbo:birthPlace`. There is also an implicit relation `dbo:HomeTown`, which is introduced by the type `dbo:Agent` of the entity `dbr:Barack`: `(dbr:Barack_Obama,rdfs:type,dbo:Agent)` and `(dbo:HomeTown, rdfs:domain, dbo:Agent)`.

We utilise both explicit and implicit relations to extract right set of relations. We first expand the potential relation candidates using explicit relations, and then further expand it with implicit relations.

Expansion 1: In this step, we fetch the property set from the instance triples in $\mathcal{A}$. To avoid large scale retrieval, we just retrieve the ontologies associated with the entity of the question rather retrieving all the ontologies of DBpedia. For each entity e in the input question, we retrieve all explicit properties from the associated ontology of this entity. We add these explicit properties to the list P1, and we call this list the *explicit property list* (EPL).

Example: Given the question "Which comic characters are painted by Bill Finger?" (a question from LC-QuAD dataset). If we do not use expansion 1, we would just get `dbo:painter` as the relation. When we apply expansion 1 to it, we also get the relation result `dbo:creator` which is derived from the explicit property list of `dbr:Bill_Finger`. As we illustrated above, `dbo:creator` is indeed the right answer.

Expansion 2: Based on Expansion 1, we add another iteration which is based on reasoning to gather the implicit property list from $\mathcal{T}$. To get the implicit property list, we first get domains and ranges from the schema $\mathcal{T}$. There are two kinds of domains and ranges. The first is global domains and ranges, and the second is the local domains and ranges. Global domains and ranges are usual

RDFS domains and ranges. In description logic form, they can be represented as $\top \sqsubseteq \forall r^{-}.GD$ (GD is a global domain of the property r), $\top \sqsubseteq \forall r.GR$ (GR is a global range of the property r), Local domains and ranges is similar but the left hand side $\top$ can be replaced by a type (such as C): $C \sqsubseteq \forall r^{-}.GD$ (GD is a local domain of the property r w.r.t. C entities), $C \sqsubseteq \forall r.GR$ (GR is a local range of the property r w.r.t. C entities). Both global and local domains and ranges can be inferred from $\mathcal{T}$. Given $\mathcal{T}$ is often fixed, so all the global and local domains and ranges can be computed offline. Given an entity e from an input question q, we add all the properties related (through global/local domains or range) to some types of e to the list P2, called the *implicit property list* (IPL).

To consider all the possibilities of relation expansion, we combine expansions of type 1 and type 2 above.

Example: Consider the question "How many shows does HBO have?". If we just use expansion 1, we will obtain the explicit property list of `dbr:HBO`. In this list, the `dbo:producer` ranks the highest. However, if we apply expansion 2 to this question, we do not only get the explicit property list of dbr:HBO, but also get the implicit property list of `dbr:HBO`. Because the `rdf:type` of `dbr:HBO` includes `dbo:Broadcaster`, and `dbo:Broadcaster` is a global range of `dbo:channel`.

It turns out that `dbo:channel` is the desired answer (and not `dbo:producer`).

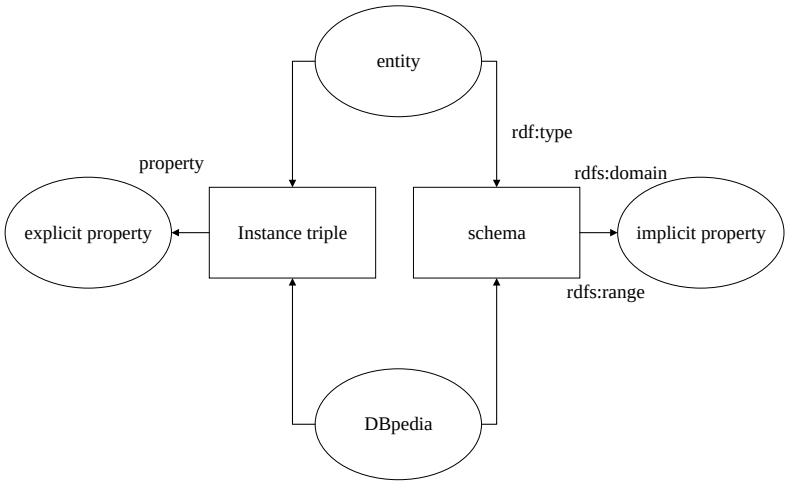

Fig. 2: The process of getting explicit property list and implicit property list

Relation Candidate Ranking Once we expand the explicit and implicit relations, we get all the possible relation candidates in the property list P. The next step for our framework is to select the best relations from these candidates which constitutes the final module of the EERL framework. In the EERL system, we reuse the approach of SIBKB to ranking candidates.

In the following, we will focus on strategies for re-ranking the candidates from the explicit property list EPL and the implicit property list IPL.

Existence Re-ranking and Extending: we perform re-ranking by extending the candidate list according to the existence of relations in the explicit property list EPL or the implicit property list IPL. From the candidate ranking step of SIBKB, we can get a K-V list. This list is ranked by the sum of similarity sum(Va). We use the principle that if relations k in relation list K matches property p in EPL or in IPL, then we will add a high weight value w1 into K-V list, which can be formalised as $R[k] = Va + w1$. If not, we will extend the K-V list by adding the property p to it with w1, which is $R[p] = w$.

LD Re-ranking: We leverage the levenshtein distance LD[11] for re-ranking and extending the candidate list. We calculate LD between extracted words from the lists EW and the words from the property candidate list PCL (both explicit property list EPL and implicit property list IPL). We restrict LD to a range (0,1). Then we identify the p in EPL and IPL with the shortest levenshtein distance to the extracted relation word EW, and give a weight value w2 to p, which is $R(p) = Va + w2$. For the weight value w2, the higher the weight value w2 the smaller the Levenshtein distance LD. Please note, for w1, w2 and w3, we define it empirically, and then through the results of the evaluation to adjust them.

Synonym Re-ranking and Extending: If the length of the extracted words string len(ew) is 1, we'll get Synonyms set S(ew) of ew first, then calculate the distance between the property candidate list PCL and s(ew) in S(ew). Similar to LD re-ranking, we restrict the distance LD in a range (0,1), identity the property p with levenshtein distance, then add a weight value w3, for the whole process $w3 = K/ld$ which can be formalised as $ld = l - distance(s(ew), p)$. $R[p] = Va + w3$.

5 Experiments

In this section, we present the experimental results to validate our approach. The open source code and evaluation results can be found at Github.[12]

5.1 Datasets

For evaluation studies, we leveraged three datasets to show the performance of EERL framework, namely the QALD datasets and LC-QuAD dataset.

[11] https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm

[12] https://github.com/zhangmeiontoweb/EERL

QALD: QALD-5[13] and QALD-7[14] are two latest benchmarking datasets from Question answering over Linked Data challenge (QALD). It mostly has simple questions (58 percent of QALD questions have a single entity and a single relation). QALD-5 have 350 questions, and QALD-7 has 215 questions.

LC-QuAD: LC-QuAD[15] has 5000 questions for QA over DBpedia and 80 percent of its questions are complex i.e. questions with more than one entity and one relation. It is a manually fully-annotated, with all keywords classified as entity or predicate, and mapped to the URIs of DBpedia. Please note, there are three kinds of questions which are not considered for evaluation: 1) the questions that are not fit for our hypothesis, i.e. questions for which relations are not the property of given entities 2) for QALD dataset, we exclude questions that don't give the SPARQL 3) in LC-QuAD dataset, the given relations for the questions which are not correct for the current DBpedia version. As we use the latest DBpedia version to retrieve the relation candidates, such questions from LC-QuAD has been ignored.

Baseline Relation Linking Tools. Various relation linking approaches have been evaluated on these datasets. SIBKB [30], ReMatch [18], and EARL[3] have been evaluated over QALD-7 and QALD-5. We therefore compare our results for the same experiment settings. We then report our results for complete LC-QuAD dataset comparing it to the baselines.

Table 1: Performance of EERL Framework Compared to Various Relation Linking Tools

QA Component	Dataset	Precision	Recall	F-Score
SIBKB	QALD-5	0.27	0.34	0.29
ReMatch	QALD-5	0.36	0.39	0.37
EARL	QALD-5	0.17	0.21	0.19
EERL	QALD-5	**0.43**	**0.49**	**0.45**
SIBKB	QALD-7	0.33	0.35	0.34
ReMatch	QALD-7	0.35	0.38	0.37
EARL	QALD-7	0.30	0.31	0.30
EERL	QALD-7	**0.42**	**0.46**	**0.43**
SIBKB	LC-QuAD	0.15	0.18	0.16
ReMatch	LC-QuAD	0.18	0.20	0.19
EARL	LC-QuAD	0.20	0.25	0.21
EERL	LC-QuAD	**0.53**	**0.58**	**0.55**

[13] https://github.com/ag-sc/QALD/blob/master/5/data/qald-5_train.json

[14] https://github.com/ag-sc/QALD/blob/master/7/data/qald-7-train-multilingual.json

[15] https://figshare.com/articles/Full_Annotated_LC_QuAD_dataset/5-782197

5.2 Experimental settings

We executed our experiments on one virtual server, with eight cores and 32 GB RAM running on the Ubuntu 16.04.3 operating system. We have reused the open source implementation of the Frankenstein Resource Platform[16] [27] and integrated our EERL framework in it for executing the different experiments. As DBpedia IRIs of the entities are also our inputs besides the natural language question, we use gold annotated linked named entities as input.

5.3 Result and Analysis

Metrics: The following evaluation metrics per relation linking approach have been used: *i*) `Micro Precision (MP)`: For a given tool, the ratio of correct answers vs. total number of answers retrieved for a particular question. *ii*) `Precision`: The average of the Micro Precision over all questions by a relation linking tool. *iii*) `Micro Recall (MR)`: The number of correct answers retrieved by a component vs. gold standard answers for given question. *iv*) `Recall (R)`: The average of Micro Recall over all questions for a given relation linking tool. *v*) `Micro F-score (MF)`: Harmonic mean of MP and MR for each question. *vi*) `F-score`: Harmonic mean of P and R for each component.

Evaluation: We evaluate our system with three metrics, they are Precision, Recall, F-score. Table 1 summarises the results of our framework compared to the baseline for the different datasets. Our framework significantly outperforms the baselines for relation linking for QALD-5 340 questions. We then extended our evaluation to QALD-7, where our framework EERL also achieves the highest performance in terms of Precision, Recall and F-score.

We then extended our performance evaluation to complex questions, and utilised the LC-QuAD dataset in two settings. In the first setting, we evaluated our performance for all the 5000 questions. We achieved significantly high Precision, Recall and F-score values for complex questions, as illustrated in Table 1. Singh et al. [31] have evaluated five relation linking approaches for 3253 questions of LC-QuAD including SIBKB, Rematch and other three tools which Frankenstein offers in its architecture. Table 1 also summarises our results compared to the best performing tool from [31], where we achieve a significant increase in performance with our EERL framework. Besides, Table 1 shows that the EERL system enhanced largely with our approach compare to all baselines across datasets.

Execution Time: Execution time is also an important KPI to evaluate our approach. Table 2 shows the execution time that each system uses for the QALD-7 and LC-QuAD datasets. The results show that our approach also significantly improves the run time. Please note, for runtime calculation, time needed for entity recognition has been cut and only the time taken by each tool to link the relation is reported in the Table 2.

[16] `https://github.com/WDAqua/Frankenstein`

Error Analysis: Even though we achieve a relative higher performance compared with state-of-the-art, we find room for improvement in our EERL framework. From the experimental results, we deduce that the errors are caused by two factors. The first factor is the effectiveness of our hypothesis. We assume that a relation in the query is the property of the entity while some of the questions' relations don't appear in the property list. And this is the main reason why our system doesn't perform better on the QALD-5 and QALD-7 datasets compared to its performance on LC-QuAD. The second reason for failure is the ranking method. We observe from the results that for many questions containing two relations, the performance of our ranking methods is limited.

Table 2: Run Time (avgerage seconds/question)

system	QALD-7	LC-QuAD
SIBKB	1.1	2.2
ReMatch	110	130
EERL	**1.3**	**1.8**

6 Discussion

From Table 1, we can infer that the SIBKB, ReMatch, and EARL systems have very limited performance. Our EERL system not only outperforms these systems, it also does not show a sharp decline in performance on complex questions. In fact, for complex questions our approach performs better than on the simple questions of the QALD dataset. One primary reason for this behaviour is the presence of more context about the entities in the complex questions, because complex questions usually contain two entities. Our approach utilizes this context to correctly predict the DBpedia relation.

Furthermore our proposed EERL framework also can give its result within a reasonable time. Our results show that, by using the property candidate list as the relation candidates, i) we can narrow the relation range and this will speed up the process of retrieving relation candidates; and ii) this approach can be used as a ranking method to rank the relation candidates to prevent filtering the correct candidates. We believe that our approach can be reused in other relation linking systems, and it could easily be extended to other knowledge graphs. This is due to the fact that other knowledge graphs have similar structure as DBpedia and they use common knowledge source (i.e. Wikipedia). Hence, our approach should work equally well for the KGs having clear separation of A box and T-Box concepts. We agree, few knowledge graphs (such as Wikidata) do not have clear and correct definition of domain and ranges, neither a well defined Ontology. Our approach will find limitation in such scenario. For knowledge graphs with

no ‘TBoxes or very incomplete TBoxes, ontology learning of domains and ranges might help, but it has been out of scope for this paper.

However, to further improve the EERL framework, we plan to optimise our approach in three possible ways. Firstly, we analysed our results, and found that over half of the wrong results were due to the wrong extracted relation words. Relation extraction from free text has been a long standing field of natural language processing research. We plan to utilise some of its techniques to extract the correct natural language label for the relation. Secondly, the similarity algorithm is a method for calculating the similarity between possible candidates and the recognised relation words. We plan to utilise external knowledge sources such as Wordnet[17] to provide a list of synonyms for relation labels. Finally, existing datasets for question answering over DBpedia do not contain large number of questions. With the availability of larger datasets, we plan to employ machine learning techniques for proposing a ranking model for the candidate relations.

7 Conclusions

In this paper, we have proposed a novel approach which can directly link the natural language relation of the question to its mention in the DBpedia. Unlike previous work in this domain, we utilize the contextual information provided by the entities in the question to find the relation in the knowledge graph. Our approach can choose the best property to match the entities by ranking the similarity between the entities' property list and extracted relation words from question. In our approach we jointly utilize the relation and property list to ensure the integrity of the question information. This has also impacted the performance, and we outperform the existing baseline approaches for relation linking. We hope, our work sets a foundation for the research community to exploit (approximate) ontology reasoning [23,21,15,22,33,13] in finding correct predicates for the questions and then applying machine learning approaches on top of it for better results. Our framework is reusable, and we have integrated it to Frankenstein framework for its reusability in creating collaborative question answering systems.

Acknowledegments

This work has been supported by the National Natural Science Foundation of China (61673304) and the Key Projects of National Social Science Foundation of China(11&ZD189)

References

1. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A Nucleus for a Web of Open Data. In ISWC, 2007.

[17] https://wordnet.princeton.edu/

2. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2003.
3. Debayan Banerjee, Mohnish Dubey, Debanjan Chaudhuri, and Jens Lehmann. Joint entity and relation linking using EARL. In ISWC, 2018.
4. Hannah Bast and Elmar Haussmann. More accurate question answering on freebase. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, pages 1431–1440, New York, NY, USA, 2015. ACM.
5. Kurt D. Bollacker, Robert P. Cook, and Patrick Tufts. Freebase: A Shared Database of Structured General Human Knowledge. In AAAI, 2007.
6. Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale Simple Question Answering with Memory Networks. CoRR, abs/1506.02075, 2015.
7. Andreas Both, Dennis Diefenbach, Kuldeep Singh, Saeedeh Shekarpour, Didier Cherix, and Christoph Lange. Qanary - A Methodology for Vocabulary-Driven Open Question Answering Systems. In ESWC, pages 625–641, 2016.
8. Zihang Dai, Lei Li, and Wei Xu. CFO: Conditional Focused Neural Question Answering with Large-scale Knowledge Bases. In ACL, 2016.
9. Biralatei J. Fawei, Jeff Z. Pan, Martin J. Kollingbaum, and Adam Z. Wyner. A Criminal Law and Procedure Ontology for Legal Question Answering. In JIST, 2018.
10. Paolo Ferragina and Ugo Scaiella. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In CIKM, pages 1625–1628, 2010.
11. D. Gerber and A.-C. N. Ngomo. Bootstrapping the linked data web. In 1st Workshop on Web Scale Knowledge Extraction@ ISWC, 2011.
12. Konrad Höffner, Sebastian Walter, Edgard Marx, Ricardo Usbeck, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. Survey on challenges of Question Answering in the Semantic Web. Semantic Web, 2017.
13. Aidan Hogan, Jeff Z. Pan, Axel Polleres, and Stefan Decker. SAOR: Template Rule Optimisations for Distributed Reasoning over 1 Billion Linked Data Triples. In Proc. of the 9th International Semantic Web Conference (ISWC2010), 2010.
14. Jiseong Kim, GyuHyeon Choi, Jung-Uk Kim, Eun-Kyung Kim, and Key-Sun Choi. The open framework for developing knowledge base and question answering system. In COLING, pages 161–165, 2016.
15. Freddy Lecue and Jeff Z. Pan. Consistent Knowledge Discovery from Evolving Ontologies. In Proc. of 29th AAAI Conference on Artificial Intelligence (AAAI-15)., 2015.
16. Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. Neural network-based question answering over knowledge graphs on word and character level. In WebConf, pages 1211–1220, 2017.
17. Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia spotlight: shedding light on the web of documents. In I-SEMANTICS, pages 1–8, 2011.
18. Isaiah Onando Mulang, Kuldeep Singh, and Fabrizio Orlandi. Matching natural language relations to knowledge graph properties for question answering. In Semantics, 2017.
19. Varish Mulwad, Tim Finin, and Anupam Joshi. Semantic message passing for generating linked data from tables. In International Semantic Web Conference, pages 363–378. Springer, 2013.
20. Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. PATTY: A taxonomy of relational patterns with semantic types. In EMNLP-CoNLL, 2012.

21. Jeff Z. Pan, Yuan Ren, and Yuting Zhao. Tractable approximate deduction for OWL. Arteficial Intelligence, pages 95–155.
22. Jeff Z. Pan and Edward Thomas. Approximating OWL-DL Ontologies. In the Proc. of the 22nd National Conference on Artificial Intelligence (AAAI-07), pages 1434–1439, 2007.
23. J.Z. Pan, D. Calvanese, Th. Eiter, Kifer M. Horrocks, I., F. Lin, and Y Zhao, editors. Reasoning Web: Logical Foundation of Knowledge Graph Construction and Querying Answering. Springer, 2017.
24. J.Z. Pan, G. Vetere, J.M. Gomez-Perez, and H. Wu, editors. Exploiting Linked Data and Knowledge Graphs for Large Organisations. Springer, 2016.
25. Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543, 2014.
26. Kuldeep Singh. Towards Dynamic Composition of Question Answering Pipelines. PhD thesis, University of Bonn, Germany, 2019.
27. Kuldeep Singh, Andreas Both, Arun Sethupat Radhakrishna, and Saeedeh Shekarpour. Frankenstein: A Platform Enabling Reuse of Question Answering Components. In ESWC, pages 624–638, 2018.
28. Kuldeep Singh, Ioanna Lytra, Arun Sethupat Radhakrishna, Saeedeh Shekarpour, Maria-Esther Vidal, and Jens Lehmann. No one is perfect: Analysing the performance of question answering components over the dbpedia knowledge graph. CoRR, abs/1809.10044, 2018.
29. Kuldeep Singh, Ioanna Lytra, Maria-Esther Vidal, Dharmen Punjani, Harsh Thakkar, Christoph Lange, and Sören Auer. QAestro - Semantic-Based Composition of Question Answering Pipelines. In DEXA, pages 19–34, 2017.
30. Kuldeep Singh, Isaiah Onando Mulang, Ioanna Lytra, Mohamad Yaser Jaradeh, Ahmad Sakor, Maria-Esther Vidal, Christoph Lange, and Sören Auer. Capturing knowledge in semantically-typed relational patterns to enhance relation linking. In K-CAP, pages 31:1–31:8, 2017.
31. Kuldeep Singh, Arun Sethupat Radhakrishna, Andreas Both, Saeedeh Shekarpour, Ioanna Lytra, Ricardo Usbeck, Akhilesh Vyas, Akmal Khikmatullaev, Dharmen Punjani, Christoph Lange, Maria-Esther Vidal, Jens Lehmann, and Sören Auer. Why reinvent the wheel: Let's build question answering systems together. In WebConf, pages 1247–1256, 2018.
32. Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In WWW, pages 697–706, 2007.
33. Edward Thomas, Jeff Z. Pan, and Yuan Ren. TrOWL: Tractable OWL 2 Reasoning Infrastructure. In the Proc. of the Extended Semantic Web Conference (ESWC2010), 2010.
34. Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs. In ISWC, pages 210–218, 2017.
35. Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over rdf data. In WebConf, pages 639–648. ACM, 2012.
36. Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Michael Röder, Daniel Gerber, Sandro Athaide Coelho, Sören Auer, and Andreas Both. AGDISTIS - Graph-Based Disambiguation of Named Entities Using Linked Data. In ISWC, pages 457–471, 2014.
37. D. Vrandecic and M. Krötzsch. Wikidata: a free collaborative knowledge base. Communications of the ACM, 57(10):78–85, 2014., 2014.