UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Faculty for Computer Science, Electrical Engineering and Mathematics
Department of Computer Science
Research Group Data Science

Seminar Report

Submitted to the Data Science Research Group
in Partial Fullfilment of the Requirements for the Degree of

Master of Science

Entity Enabled Relation Linking

by
HARSHITH SRINIVAS

(MATRICULATION NUMBER : 6896703)

Seminar Supervisor:
Prof. Dr. Muhammad Sherrif

Paderborn, February 5, 2021

Abstract.

In knowledge graph, important aspect to keep focus is on Relation linking. In general terms it means that we have to identify relevant relations from given Natural language question using knowledge graph. Since existing approaches for entity extraction and linking are more efficient when compared to Relation linking, the main idea in this paper is to make use of entities extracted from natural language input question to support relation linking.To achieve this, we put forward a novel approach, based on DBpedia entities, to compute relation candidates. And also, we have evaluated our approach based on various existing baseline systems in both recall, precision and run-time.

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

Contents

0.1 Introduction

In recent years, many Knowledge Graphs (KG) [PZS$^+$19] in Semantic Web has kept increasing in terms of both numbers and size. Most popular ones among them are DBpedia, YAGO, Freebase and Wikidata. Many question answering (QA) systems have been implemented to gather information easily from these KGs. Researchers in this domain follow two main approaches to address question answering systems over KG. First approach developed by these researchers is end-to-end question answering systems on KGs which has large training data in benchmarks (example: Freebase) and use Deep Learning and Machine Learning techniques to predict mapping and linking of entity and relation linking in the natural language input question to their KG occurrences to extract answers. Since, there is large amount of training data, Freebase Knowledge graph is used in these types of question answering systems(e.g th SimpleQuestion benchmark for freebase would contain 100,000 questions). But, DBpedia has limitans at most 5000 questions. So later approach is on KGs which has limited training data by using semantic parsing to build question answering systems which relies on semantics related to natural language to comprehend the natural language input question. Semantic parsing based question answering (SQA) systems carry out series of tasks (often referred to QA pipeline) to convert natural language questions to their corresponding SPARQL query These systems implement independent component(s) in the entity and relationship linking architecture through DBpedia. That is, to connect the entities and relationships extracted from the input question to their occurrences in the knowledge graph. Most QA systems face the following challenges when doing so: 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 question answering systems that facilitate reusability of QA components.

In this paper, we discuss the problem of relation linking to develop SQA systems in a collaborative effort, Many frameworks, such as Qanary, OKBQA6 and Frankenstein, are built by reusing existing independently released tools that use modular approaches for building QA systems.A variety of different entities and relation linking tools, such as DBpedia Spotlight systems, AGDISTIS, SIBKB, ReMatch and TagMe are reused.In this paper, we will create a new relation linking component following this approach, embed it in an existing framework (in our case, Frankenstein) and compare its efficiency against the state of the art.We concentrate on relation linking since independent individual linking tools already perform well when applied to QA systems such as Frankenstein, whereas all existing independent relation linking tools, on the other hand, fail miserably in terms of precision and runtime.The overall efficiency of the QA systems is influenced by this failure of this relation linking tools.Recently released studies by Singh et. al. has argued that one of the key reasons behind the limited success of the relation linking tool is that the current relation linking tools concentrate more on the recognition of relations in the actual question, while totally missing the context of the entities co-occurring with these relations.

Therefore, in this article, we plan to make use of entities occurring in questions to assist the relation linking role of the DBpedia Knowledge Graph.More specifically, properties that are logically related to the target entities (as domains or ranges) are referred to as the candidate property list (or simply property list). This list of properties will then be used to extend the set of relation candidates that can be used to construct SPARQL queries in the QA pipeline. Our tests later in this paper would demonstrate that the use of logically connected property candidates contributes to major improvements not only in recall, accuracy, but also in runtime.

For an instance, given an natural language input questions “Which comic characters are painted by Bill Finger?”, we can extract the entity phrase "Bill Finger" and relation phrase "Painted by". Typically, existing relation linking approaches extract the relation phrase in the question directly, while ignoring the entities (in this case, Bill Finger) and expanding the relation candidate using a synonym list like "painter". When tools like these try to map the relation candidate to DBpedia relations we get dbo:painter as an answer to the resulting SPARQL query.And this SPARQL query returns NULL because dbo:painter is not the right property of the entity Bill Finger.Instead, we believe that the entity Bill Finger is already connected to its DBpedia mention and acts as one of the inputs in addition to the natural language question.Property list is created by gathering all the DBpedia entity properties($dbr : Bill\_Finger$), that also has property names that has $dbr : Bill\_Finger$ as domains or ranges. After this we utilize property list, which also includes ranking's of the list and be assure that chosen range properties are compatible with $comiccharacters$. Finally, in the end we get $dbo : creator$ as relation candidate. We also can conclude that $dbo : creator$ is the best answer, when we apply the same result to SPARQL query corresponding to input question.By following this we eliminate the requirement of large training data by focussing on DBpedia structure (which includes entities and there properties), and keeping context of relation having entities in question.

Keeping this as an 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 are summarized as follows: first, a novel approach to the generation and ranking of candidate relationships used in QA systems. Second, the application of this strategy in the Entity Enabled Relation Linking(EERL). Third, an in-depth evaluation of this approach to databases with 5000 different questions, resulting in an improvement over the present state-of-the-art baseline approach.

As much of the current works in the literature, we are checking our contribution to DBpedia. However, there is no clear presumption in our work on the structure or schema of the underlying knowledge graph, and our approach should be similarly valid and can be applied to every other knowledge graph.

0.2 Background

0.2.1 Knowledge Graph

Given "data sub-graph" $\mathcal{A}$ and "schema sub-graph" T, we define knowledge graph as $\mathcal{G} = \tau \cup \mathcal{A}$. Facts in ABox is represented as triples in the following two ways:

– – $Relation\, assertion(h, r, t)$, where $h(t)$ is the head (tail) and r is the relation. For an instance, e. g.(dbr:Barack_Obama, dbp:birthPlace, dbr:Hawaii)

– – $Type\, assertion(e, rdf\ :\ type, C)$, where e is an entity, $rdf\ :\ type$ is the instance of relation with the standard W3C RDF specification and C is a type. For an instance, e. g.(dbr:Bill_Finger, rdf:type, dbo:Person)

A TBox represents type inclusion axioms, such as $(dbo\ :\ Person\, rdfs\ :\ subClassOf\, dbo\ :\ Agent)$, and relation axioms, such as $(dbp\ :\ birthPlace\, rdfs\ :\ domains\, dbo\ :\ Person)$ and $(dbp\ :\ birthPlace\, rdfs\ :\ range\, dbo\ :\ Person)$. Also, there can be various type and relation axioms as stated in W3C standard knowledge grapgh schema language OWL, that is based on a descriptive logic.

In rest of paper, we make use of $E(\mathcal{A})$ $(resp.\ R(\tau))$ which refers set of entities (resp. Relations) in $\mathcal{A}$ $(resp.\ (\tau))$.Note that the set of relations in $\mathcal{A}$ is a subset of the set of relations in τ , some of which might not have instances in $\mathcal{A}$.

0.2.2 Problem Statement

Firstly, let us define the problem of relation linking for factoid questions, before proposing our new research problem. Given the schema τ of a knowledge graph $\mathcal{G} = \tau \cup \mathcal{A}$ and an input natural language question q, the task of entity linking is to find the set of relations $R_q \subseteq R(\tau)$ 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 explored within these questions. Formally, given a knowledge graph $\mathcal{G} = \tau \cup \mathcal{A}$, an input natural question q and a set of entities $E_q \subseteq E(\mathcal{A})$ identified in q, the task of entity enabled relation linking is to identify a set of relations $R_q \subseteq R(\tau)$ for the set of relation phrases in q based on the entities E_q.

Notice that the ABox $\mathcal{A}$ of $\mathcal{G}$ entities, as well as their interrelations, are not taken into account in the (pure) relation linking task, but in the entity-enabled relation linking, which, in effect, often takes into account the implicit connections between the entities in $\mathcal{A}$. This makes an entity's enabled relation linking a (much) more complicated issue than a (pure) relation linking.

0.3 Related Work

In the light of the natural language input question and the knowledge graph, the role of the relation linking is to define the appropriate relationship in the knowledge graph for the relation phrases of the input question. There are a number of DBpedia-linked relation tools and systems available.

PATTY [NWS12]: *PATTY* is a great resource for textual patterns denoting binary relationships between entities. It is a two-column large knowledge resource, with one column representing natural language relation patterns and the other column containing associated DBpedia predicates. However, *PATTY* cannot be used directly as a component for linking relations in a QA system and must be modified according to the requirements of individual developers
BOA: *BOA* can be used to extract natural language representations of predicates independent of the language when a Named Entity Recognition service is provided. Like *PATTY*, *BOA* must be modified before direct use in a QA system.
SIBKB: *SIBKB* provides search 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 words with DBpedia predicates for an independent relation linking tool that accepts questions as input and returns DBpedia properties as output.
ReMatch: *ReMatch* system is an independently reusable tool to match natural language relations with KB properties. This tool uses dependency parsing features with matching rules and then performs matching against KG properties extended with the lexicon Wordnet. However, the runtime for each query is relatively slow.
EARL: *EARL* is the latest approach to linking entities and relations together. This tool treats entity and relation linking as a single step. First, it aims to identify entities in the query and identifies the relation associated with the entities following a graph traversal approach. EARL determines the best semantic link between all keywords in the question by exploiting the link density between entity and relation candidates.

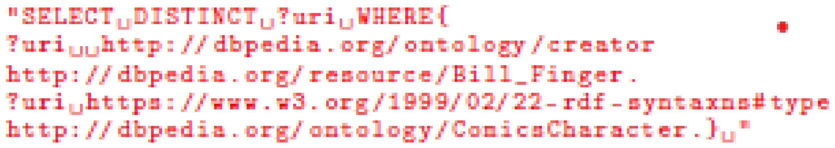
The work of Usbeck et al. proposes an entity linking tool *AGDISTIS* that is closest to our approach. *AGDISTIS* combines the algorithm *HITS* with label expansion strategies and string similarity measures. Similar to our approach, where we rely on linked URIs of entities in a question as input in addition to the question, *AGDISTIS* accepts a natural language question (or sentence) and recognised entities as input to provide disambiguated entity URIs. However, unlike our approach, it is limited to entity linking and does not perform relation linking. Moreover, *TBSL*'s QA system uses entities in the query Q to generate templates that are later populated with properties from the graph, which is quite similar to our idea of using entities in finding the correct predicate. *TBSL* uses external resources *BOA*, in addition to string matching, to find the correct relations. Our approach goes a step further and relies heavily on the ontology to find the correct predicates for the given entity without using any external knowledge resource. This demonstrates the power of using knowledge encoded in the knowledge graph itself. Furthermore, the use of entities to map relations in a question is new to QA relation linking, but this has been well explored in ontology mapping (alignment).
In mapping tables to ontologies, for example, there are approaches that create a candidate list of properties and arrange them according to how the entities are linked in the table cells. There are also efforts to develop rich QA, e.g., in the legal domain.

0.4 Approach

In this section, We will discuss how entities E_q is used in relation linking

0.4.1 Preliminaries and Proposed hypothesis

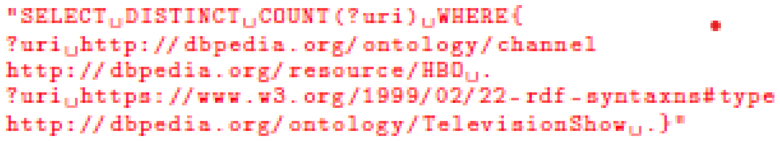
To evaluate our new approach to solving the relation linking problem, we analysed 100 randomly selected question-answer pairs from the benchmarking datasets. Analysing the SPARQL query associated with the input questions, we find that most of the predicates of these queries (i.e. the KG relations for the natural language relations occurring in the input question) are the properties of the entities in the questions. For example, given input question **"Which comic characters are painted by Bill Finger?"** (A question chosen from LC-Quad dataset), SPARQL query for the given 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.}␣"
```

Query 1

In this query, the predicate $dbo : creator$ of the associated entity $dbr : Bill_Finger$ is one of the properties of $dbr : Bill_Finger$ in DBpedia. Moreover, it is often the case that there is no natural language label of a relation in the query. For an instance, the input question **"How many shows does HBO have?"** (A question chosen from LC-Quad dataset) has no natural langaugae relation label. These kinod of questions are called as questions with *hidden relations*. SPARQL query for 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␣.}"
```

Query 2

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

Hypothesis: Based on above analysis, we have proposed 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 our knowledge this simple hypothesis has not yet been exploited in any of the current state-of-the-art approaches to QA relation linking.

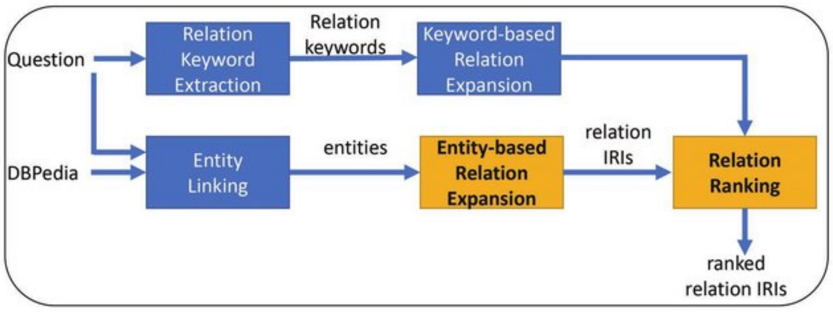


Fig 1 : Conceptual Architecture of our EERL Framework. All of the baseline relation linking frameworks found in the literature share the top part of the pipeline. Our contribution in the EERL framework is the additional entity-based part of the pipeline in the bottom half of the image.

Based on this approach, they have developed an approach that follows five main steps (see Figure 1): 1) Relation Keyword Extraction 2) Keyword based Relation Expansion 3) Entity Linking 4) Entity-Based relation Expansion 5) Relation Linking

0.4.2 EERL Framework

As illustrated in figure 1, our EERL framework has five steps/modules. EERL framework has two inputs, a natural language input question $_q$ and the DBpedia knowledge graph.

Relation Keyword Extractor: In this step we extract the relation keywords from the given natural language input questions.

Example: In the question **"Which comic characters are painted by Bill Finger?"**, we extract the **"painted by"** phrase. We utilize $TexRazor\ API$ which provides us with relation phrases.

Keyword based Relation Expansion: In this step we expand the extracted relation keywords using background knowledge.

Example: The relation phrase **"painted by"** is expanded by using knowledgebase $PATTY$ to get corresponding relation phrases. we first convert **"painted by"** in vector using *Glove* and then use this vector form of $PATTY$ to extract most suitable relation phrase. For the above example, we get “painter” as an answer.

These two steps are performed by all or most base systems, and we do not raise them as a novelty. We list them only for completeness and as part of the overall solution provided. In a parallel step, the DBpedia IRIs of the entities from the question (in this case $dbr : Bill_Finger$) are used to create a property list as described below..

Entity based Relation Expansion: This step is core of our implementation which relies on our proposed hypothes. In this step we link entites to DBpedia IRIs.

Given a KG $\mathcal{G} = \tau \cup \mathcal{A}$, the entities in a knowledge graph are the nodes of $\mathcal{G}$. These nodes are connected to other nodes (i.e., other entities) by 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 use both explicit and implicit relations to extract the correct set of relations.

Expansion 1: In this step, the property set is retrieved from the instance triples in $\mathcal{A}$. To avoid extensive retrieval, only the ontologies associated with the entity of the question are retrieved, and not all the ontologies of DBpedia.For each entity e in the input question, all the explicit properties are retrieved from the associated ontology of that entity. We add these explicit properties to the list $P1$ and call this list as "explicit property list" (EPL).

Example: Given the question **"What comic characters are drawn by Bill Finger?"** (a question from LC-QuAD dataset). If we don't use expansion 1, we would only get $dbo : painter$ as a relation. If we apply extension 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 have shown above, $dbo : creator$ is indeed the correct answer.

Expansion 2: Based on extension 1, we add another iteration based on reasoning to obtain the implicit property list of τ . To obtain the implicit property list, we first get domains and ranges from the schema τ . There are two types of domains and ranges. The first type is global domains and ranges and the second type is local domains and ranges. Global domains and ranges are the usual RDFS domains and ranges.

In description logic form, they are represented as, $\top \subseteq \forall r^{-}.GD$ where GD is global domain of property r and $\top \subseteq \forall r.GR$ where GR is global domain of property r. Local domains and ranges are also similar to global but in left hand side we have type C instead of $\top$ as, $C \subseteq \forall r^{-}.GD$ where GD is a local domain of property r with respect to C and $C \subseteq \forall r^{-}.GR$ where GR is a local range of property r with respect to C. Both global and local domains and ranges can be derived from τ. Since τ is often fixed, all global and local domains and ranges can be computed offline. Given an entity e from an input query q, we add all properties associated (by global/local domains or ranges) with some types of e to the list $P2$, called the "implicit property list" (IPL).

Example: Consider the question "How many shows does HBO have". Using only *expansion* 1, we get the explicit property list of $dbr : HBO$. In this list, the $dbo : producer$ is in first rank. However, if we apply *expansion* 2 to this question, we get not only the explicit property list of $dbr : HBO$, but also the implicit property list of $dbr : HBO$. This is because the $rdf : type$ of $dbr : HBO$ contains $dbo : Broadcaster$, and $dbo : Broadcaster$ is a global range of $dbo : channel$.Even here we get to know that $dbo : producer$ is not correct answer and $dbo : channel$ is the right answer.

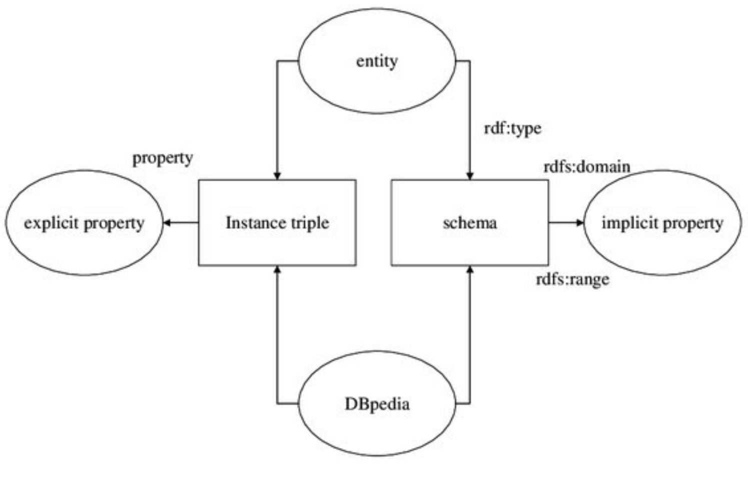


Fig. 2: The process of getting explicit property list and implicit property list

Relation Candidate Linking: After expanding the explicit and implicit relations, we obtain all possible candidate relations in the property list P. The next step for our system is to select the best relations from these candidates, which is the final module of the EERL system. In the EERL system, we use the $SIBKB$ approach for ranking the candidates.
We have make used of three different strategies in this step. which is explained below:

Existence Re-ranking and Extending: We perform a new re-ranking by expanding 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 obtain a $K - V$ list. This list is ranked according to the sum of similarity $sum(V_a)$. Then, we make use of principle that if relations k in relation list K do match with property p in EPL or in IPL, Later we add height-weight value wl into $K - V$ list. Hence we get, $R[k] = V_a + wl$. If not, we expand the property $K - V$ list by adding property p with $w1$, where we get $R[p] = w$

LD Re-ranking: In this step, we make use of Levenshtein distance LD to re-rank and extend the list of candidates. LD is calculated between words extracted from list EW and the words from procperty candidate list PCL from both IPL and EPL. Then we identify the p in EPL and IPL with the shortest Levenshtein distance to the extracted relation word E_W and give p, a weight value $w2$, which is $R(p) = V_a + w2$. For the weight value $w2$, the higher the weight value $w2$, the smaller the Levenshtein distance LD. Note that for $w1$, $w2$, and $w3$, we set them empirically and then adjust them by the results of the evaluation.

Synonym Re-ranking and Extending: We will get Synonyms set $S(ew)$, if the extracted words string $len(ew)$ is 1. Then we calculate the distances between PCL and $s(ew)$ of $S(ew$. Even here, we restrict the distance in range of $(0, 1)$. Identity the property p with levenshtein distance, then add a weight value $w3$, for the whole process $w3 = K \sim ld$ which can be formalised as $ld = l\,v\,distance(s(ew),\, p)$. $R[p] = V_a + w3$.

0.5 Experiment

0.5.1 Data-sets

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

QALD: The two bench-marking data sets from Question Answering over Linked Data challenge ($QALD$) are $QALD-5$ and $QALD-7$. It mostly has simple questions and 58 % of $QALD$ dataset has single entity and relations. $QALD-5$ has 350 questions and $QALD-7$ has 215 questions.

LC-QuAD: $LCQuAD$ has 5000 simple questions for QA over DBpedia and 80 % of the questions are complex, that is, questions with more than one entity and one relation. And, Also it is manually annotated with all keywords classified as entity and predicate and mapped to the URIs of DBpedia.
Please note that there are three types of questions that will not be considered for scoring: 1) the questions that do not fit our hypothesis, i.e. questions where the relations are not the property of the given entities 2) for the QALD dataset, we exclude the questions that do not give SPARQL 3) in the LC -QuAD dataset, the given relations for the questions that are not correct for the latest DBpedia version. Since we use the latest DBpedia version to retrieve the relation candidates, such questions from LCQuAD were ignored.
Baseline Relation Linking Tools. Several relationship linking approaches were evaluated on these datasets. $SIBKB$, $ReMatch$ and $EARL$ were evaluated over $QALD-7$ and $QALD-5$. We therefore compare our results for the same experimental settings and then report our results for the full $LC-QuAD$ data set compared to the baselines.

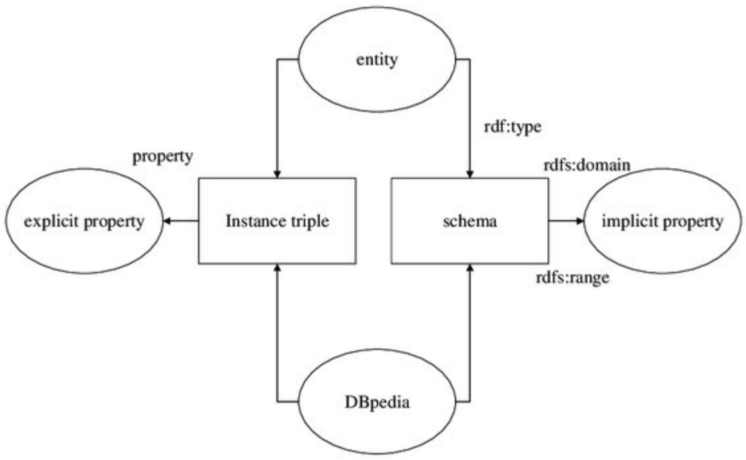


Table 1: Performance of EERL Framework Compared to Various Relation Linking Tools

0.5.2 Experiment settings

We ran our experiments on a virtual server with eight cores and 32 GB of RAM running the Ubuntu 16.04.3 operating system. To run the various experiments, we reused the open-source implementation of Frankenstein Resource Platform [[SBRS18]] and integrated our EERL framework into it. Since DBpedia IRIs of entities are our inputs in addition to the natural language question, we use gold-annotated linked named entities as inputs.

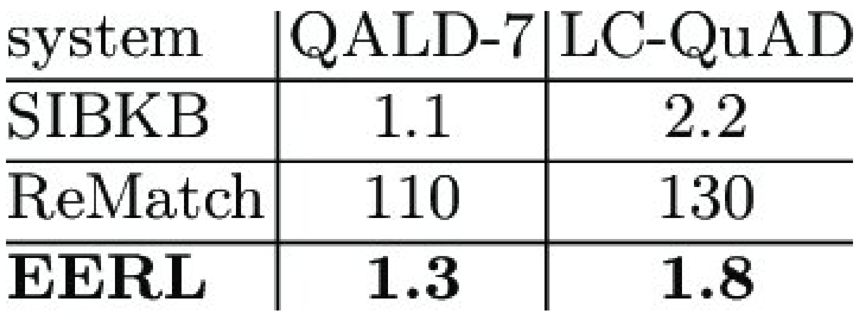
0.5.3 Result and Analysis

Metrics: The following evaluation metrics per relationship linking approach were used: i) Micro Precision (MP): for a given tool, the ratio of correct answers compared to the total number of answers retrieved for a given question. ii) Precision: the average of Micro Precision across all questions by a relationship linking tool. iii) Micro Recall (MR): The number of correct answers retrieved from a component compared to the gold standard answers for a given question. iv) Recall (R): The average of Micro Recall across all questions for a given relationship 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 test our method with three metrics: Precision, Recall, F-score. Table 1 outlines the findings of our system compared to the standard for the different datasets. Our system greatly exceeds the baselines for the relationship between $QALD-5$ 340 problems. We then expanded our analysis to $QALD-7$, where our EERL architecture still achieves the best value in terms of Precision, Recall and F-score.

We then extended our performance assessment to complex questions using the $LC-QuAD$ dataset in two settings. In the first setting, we evaluated our performance for all 5000 questions. We achieved significantly high Precision, Recall, and F-Score values for complex questions, as shown in Table 1. Singh et al. [[SRB$^+$18]] evaluated five relation linking approaches for 3253 questions from $LC-QuAD$, including $SIBKB$, Rematch and three other tools offered by Frankenstein in its architecture. Table 1 also summarises our results compared to the best performing tool from, where we achieve a significant performance improvement with our EERL framework. In addition, Table 1 shows that our approach greatly improved the EERL system compared to all baselines across all datasets.

Execution Time: Execution time is also an important KPI for evaluating our approach. Table 2 shows the execution time required by each system for the $QALD-7$ and $LC-QuAD$ datasets. The results show that our approach also significantly improves the execution time. Note that for the runtime calculation, the time required for entity detection has been truncated and only the time required by each tool for relation linking is reported in Table 2.



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

Table 2: Run Time (avgerage seconds/question)

0.6 Discussion

From Table 1, we can infer that the SIBKB [DBLP:conf/kcap/SinghMLJSV0A17], ReMatch[[MSO17]], and EARL[[BDCL18]] systems have limited performance. Our EERL system not only outperforms these systems, it also does not show a sharp drop in performance on complex questions. In fact, our approach performs better on complex questions than on the simple questions in the QALD dataset. A major reason for this behavior is the presence of more context about the entities in the complex questions, since complex questions typically contain two entities. Our approach uses this context to correctly predict the DBpedia relation.
Our results show that by using the property candidate list as relationship candidates, we can i) narrow down the relationship domain and this speeds up the process of retrieving relationship candidates; and ii) this approach can be used as a ranking method to rank the relationship candidates to prevent filtering out the wrong candidates. Also, We accept that some knowledge graphs (such as Wikidata) do not have a simple and accurate description of the domain and range, nor do they have a well-defined ontology. In such a situation, our solution would face limitations.
However, to further improve the EERL framework, we plan to optimise our approach in three ways. First, we analysed our results and found that over half of the incorrect results were due to the incorrectly extracted relation words. Relation extraction from free text is a long-standing area of research in natural language processing. We plan to use some of these techniques to extract the correct natural language term for the relation. Second, the similarity algorithm is a method for computing the similarity between possible candidates and the recognised relation words; we plan to use external knowledge sources such as Wordnet[[MJGB11, **?**]] to provide a list of synonyms for relation labels. Finally, the existing datasets for answering questions on DBpedia do not contain a large number of questions. With the availability of larger datasets, we plan to use machine learning techniques to propose a ranking model for candidate relations.

0.7 Conclusion

In this work, we proposed a novel approach that can directly link the natural language relation of the question to its mention in DBpedia. Unlike previous work in this area, we use the contextual information provided by the entities in the question to find the relation in the knowledge graph. Our approach can select the best property for the entities by evaluating the similarity between the entities' property list and the extracted relation words from the question. In our approach, we use the relation and the property list together to ensure the integrity of the question information. This has also had an impact on performance, and we outperform existing baseline approaches for relation linking. We hope that our work lays a foundation for the research community to leverage (approximate) ontology reasoning [[PCE+17, PRZ16, LP15]] in finding correct predicates for questions, and then build on that to apply machine learning approaches to achieve better results. Our framework is reusable, and we have integrated it with the Frankenstein framework so that it can be reused to build collaborative question answering systems.

Bibliography

[BDCL18] Debayan Banerjee, Mohnish Dubey, Debanjan Chaudhuri, and Jens Lehmann. Joint entity and relation linking using EARL. In Marieke van Erp, Medha Atre, Vanessa López, Kavitha Srinivas, and Carolina Fortuna, editors, *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018*, volume 2180 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.

[LP15] Freddy Lécué and Jeff Z. Pan. Consistent knowledge discovery from evolving ontologies. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 189–195. AAAI Press, 2015.

[MJGB11] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In Chiara Ghidini, Axel-Cyrille Ngonga Ngomo, Stefanie N. Lindstaedt, and Tassilo Pellegrini, editors, *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, ACM International Conference Proceeding Series, pages 1–8. ACM, 2011.

[MSO17] Isaiah Onando Mulang, Kuldeep Singh, and Fabrizio Orlandi. Matching natural language relations to knowledge graph properties for question answering. In Rinke Hoekstra, Catherine Faron-Zucker, Tassilo Pellegrini, and Victor de Boer, editors, *Proceedings of the 13th International Conference on Semantic Systems, SEMANTICS 2017, Amsterdam, The Netherlands, September 11-14, 2017*, pages 89–96. ACM, 2017.

[NWS12] Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. PATTY: A taxonomy of relational patterns with semantic types. In Jun'ichi Tsujii, James Henderson, and Marius Pasca, editors, *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1135–1145. ACL, 2012.

[PCE$^+$17] Jeff Z. Pan, Diego Calvanese, Thomas Eiter, Ian Horrocks, Michael Kifer, Fangzhen Lin, and Yuting Zhao, editors. *Reasoning Web: Logical Foundation of Knowledge Graph Construction and Query Answering - 12th International Summer School 2016, Aberdeen, UK, September 5-9, 2016, Tutorial Lectures*, volume 9885 of *Lecture Notes in Computer Science*. Springer, 2017.

[PRZ16] Jeff Z. Pan, Yuan Ren, and Yuting Zhao. Tractable approximate deduction for OWL. *Artif. Intell.*, 235:95–155, 2016.

[PZS+19] Jeff Z. Pan, Mei Zhang, Kuldeep Singh, Frank van Harmelen, Jinguang Gu, and Zhi Zhang. Entity enabled relation linking. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, volume 11778 of *Lecture Notes in Computer Science*, pages 523–538. Springer, 2019.

[SBRS18] Kuldeep Singh, Andreas Both, Arun Sethupat Radhakrishna, and Saeedeh Shekarpour. Frankenstein: A platform enabling reuse of question answering components. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 624–638. Springer, 2018.

[SRB+18] 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 Pierre-Antoine Champin, Fabien L. Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, editors, *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1247–1256. ACM, 2018.