Leveraging Semantic Parsing for Relation Linking over Knowledge Bases

Nandana Mihindukulasooriya(✉), Gaetano Rossiello, Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Mo Yu, Alfio Gliozzo, Salim Roukos, and Alexander Gray

IBM Research, T.J. Watson Research Center, Yorktown Heights, NY, USA
{nandana.m,gaetano.rossiello,ibrahim.abdelaziz1,
srini,alexander.gray}@ibm.com,
{kapanipa,yum,gliozzo,roukos}@us.ibm.com

Abstract. Knowledge base question answering systems are heavily dependent on relation extraction and linking modules. However, the task of extracting and linking relations from text to knowledge bases faces two primary challenges; the ambiguity of natural language and lack of training data. To overcome these challenges, we present SLING, a relation linking framework which leverages semantic parsing using Abstract Meaning Representation (AMR) and distant supervision. SLING integrates multiple approaches that capture complementary signals such as linguistic cues, rich semantic representation, and information from the knowledge base. The experiments on relation linking using three KBQA datasets, QALD-7, QALD-9, and LC-QuAD 1.0 demonstrate that the proposed approach achieves state-of-the-art performance on all benchmarks.

Keywords: Relation linking · Semantic parsing · Knowledge bases · Question answering

1 Introduction

Relationship Extraction and Linking (REL) is a necessary task for Knowledge Base Question Answering (KBQA) [20–22]. The goal of REL in KBQA is to identify the relations in input natural language questions and link them to their equivalent relations in a knowledge base, which are then used to construct the corresponding SPARQL query to retrieve answers. For example, we show below the corresponding DBpedia [9] SPARQL query for the question "Who is starring in Spanish movies produced by Benicio del Toro?":

```
SELECT DISTINCT ?result WHERE {
    ?film dbo:starring ?result .
    ?film dbo:country dbr:Spain .
    ?film dbo:producer dbr:Benicio_del_Toro .
    }
```


J. Z. Pan et al. (Eds.): ISWC 2020, LNCS 12506, pp. 402–419, 2020.
https://doi.org/10.1007/978-3-030-62419-4_23

Identifying the relevant relations in the question and linking them to their equivalent DBpedia relationships `dbo:starring`, `dbo:country`, and `dbo:producer` is the primary goal of REL in the context of KBQA.

REL for KBQA faces the following challenges: (1) Knowledge bases such as DBpedia, Wikidata, and Freebase have a large number of relationships which makes it challenging to acquire training data to build data-intensive deep learning models. For instance, DBpedia has thousands of relationships (some of which are generated automatically from *Wikipedia* infobox keys). (2) There is an extensive lexical gap between the surface form of relations in text and how they are represented in the KB, which makes the linking between them challenging. For example, the question above does not explicitly mention any reference to the relationship `dbo:country` which is a required relation to form the SPARQL query that can retrieve the answer. (3) Determining multiple relationships and their source and target concepts in a sentence. The example question above requires three relationships to be linked with their corresponding source and target entities/unbound variables.

In order to address the aforementioned challenges, in this work, we propose our `Semantic LINkinG` system: `SLING`; a distant supervision based approach that leverages semantic parsing such as Abstract Meaning Representation (AMR) for relation extraction and linking. Distant supervision techniques address the challenge of lack of training data, particularly for thousands of relations in KBs such as DBpedia. Transforming the text to a semantic parse such as AMR, provides advantages that include (1) normalising relations to a set of standard `PropBank` predicates, (2) identification of named entities, and (3) entity typing with a predefined type system. These characteristics of AMR help to alleviate the lexical gap by reducing different phrasings of relations to its predicate set. Furthermore, they also help to automatically determine the relationship structure of an input question and extract all relationships useful for forming a SPARQL query, hence addressing the challenge of extracting multiple relationships from questions text.

In summary, the main contributions of this paper are as follows:

- A generic framework integrating different approaches for REL based on statistical predicate alignment, word embedding and neural networks. Furthermore, the framework is modular to allow for integrating more techniques to the pipeline.
- A novel approach that harnesses AMR semantic parses of texts for REL in KBQA. Our novel usage of AMR successfully addresses the lexical gap and multiple relationship problems in REL, and achieves the new state-of-the-art on multiple benchmarks (QALD [21,22] and LC-QuAD 1.0 [20]).
- A distant supervised technique that can generate mappings between text, AMR, and KB relations leveraged for training relation classification models in the absence of task-specific training data.

The rest of the paper is organised as follows: In Sect. 2, we position our work compared to related work in REL, and Sect. 3 provides an overview of the proposed approach including a summary of each sub-module. In Sect. 4 we describe the metadata generated from the question to be used by the relation linking

modules. Section 5 describes how the distant supervision data is generated and the two relation linking modules that leverage these data to generate relation linking candidates. Section 6 describes how SLING aggregates the scores from the different relation linking modules and identify the top-k relations. Next, we evaluate our system in comparison with state-of-the-art relation extraction linking approaches in Sect. 7. Finally, we conclude and present our future work in Sect. 8.

2 Related Work

Several KBQA systems have been proposed in the literature which differ according to the traits of the datasets used, such as the amount of training data available, the complexity of the questions, or if the formal queries are provided as ground truth [2,5]. In most KBQA systems, extracting relations from the questions and linking them to the KB is an essential step to generate the structure of the formal queries.

REL has been addressed using deep learning models for KBQA datasets with large number of training examples. These deep learning approaches fall into two main categories: classification-based models [10] and ranking-based models [24]. However, there are drawbacks using end to end neural approaches for REL linking: (1) they are limited only to the questions expressing one single relation in the KB. (2) they cannot be applied in the case of a lack of training data.

QALD [21,22] and LC-QuAD [20], are well-known datasets derived from DBpedia [1], represent a real-world evaluation benchmark in evaluating KBQA systems. The limited amount of training examples of these datasets along with the complex questions involving an arbitrary number of relation types make the task of identifying and linking relations significantly challenging. For KBQA, this is addressed either as a part of an end-to-end question answering system such as GAnswer [6] or by training a model to select the best off the shelf REL system Frankenstein [17]. Our work focuses on building such off the shelf tools, particularly for KBQA.

There are four primary REL works that are geared towards the above mentioned KBQA datasets [4,12,16,18]. ReMatch [12] models every KB relation into a data structure that encapsulates the relation and some enhanced attributes from dependency parsers and WordNet taxonomy. It then applies a number of similarity measures between the question and the KB relations to output a list of candidate relations. EARL [4] jointly links relation and entities from natural language to KGs. It extracts the keywords from the question, identifies them as entity or relation, then gets a list of candidates from the KG. Similarly, Falcon [16] is an approach that jointly links entities and relations in question-like sentences to DBpedia. For a given question, it applies a number of steps to extract candidate entities and relations including POS tagging, tokenization, compounding and n-gram tiling. Falcon is the state-of-the-art approach for REL on the QALD-7 and LC-QuAD 1.0 datasets. Entity Enabled Relation Linking (EERL) [14] introduces entity-based relation expansion to the existing commonly

used keyword based relation extraction with the hypothesis that relations that occur in the question should be either properties of the entities in question or of their types.

However, none of the above mentioned REL methods for KBQA have explored the use of semantic parsers, whilst our work is the first to leverage the AMR of the question text as one of the inputs in an effort to reduce the ambiguity of natural language. Furthermore, we train a distantly supervised neural model in order to address the lexical gap issue between the relations expressed in the questions and the relation labels in the KB. This is inspired by the use of distant supervision for standard relation extraction tasks when there is limited or no training data for the target relations [15].

3 System Overview

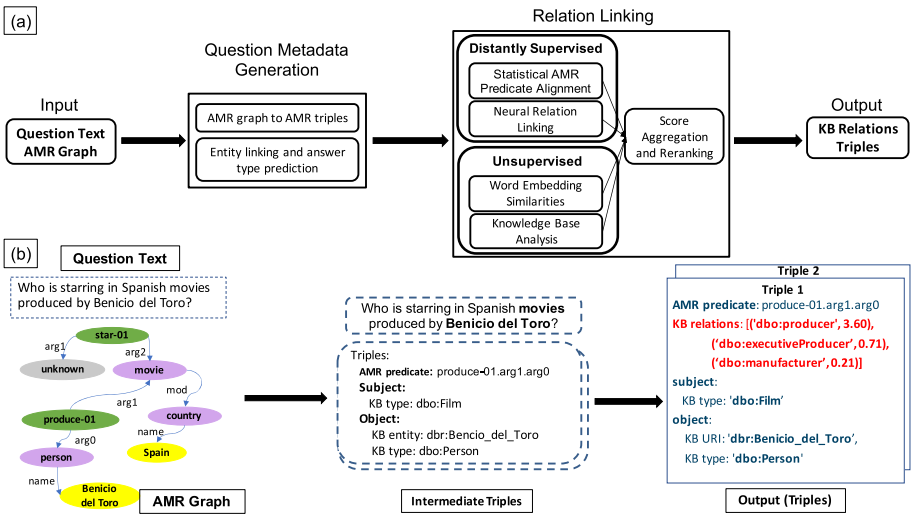

Fig. 1. (a) Overview of `SLING`. (b) Example driven flow of the approach.

Figure 1 shows an overview of `SLING` with 1-(a) showing a process-oriented view while 1-(b) illustrating with an example. The input to `SLING` is a question in natural language along with its corresponding AMR representation. The required output is a ranked list of relations corresponding to every subject-object pair in the sentence. The input is processed by the components in *Question Metadata Generation* (Sect. 4) to extract AMR triples (subject-object pair and their AMR predicate) and generate metadata corresponding to each of them. Each module in *Relationship Linking* produces a ranked list of KB relations with scores for

a metadata-enriched AMR triple. These are aggregated to produce the required output. The source code is available at GitHub[1] under Apache 2.0 license.

SLING's design is modular to allow different relation linking modules to be plugged in and used as needed. The motivation for using multiple modules is to capture different signals such as linguistic cues from the question, richer semantic information from the AMR predicates and roles, semantic similarities of terms, and heuristics from the KB itself.

We have implemented four different relation linking modules. The first two are novel relation linking approaches; both rely on distantly supervised data which we create automatically using the DBpedia and Wikipedia documents (see Sect. 5). The other two relation linking modules are unsupervised (see Sect. 6). Each of the four modules provides relations with corresponding scores. We aggregate these scores to output a final ranked list of relations.

An example of the metadata and the output is shown in Fig. 1 (b). The input data includes the question text and its AMR graph. The modules in *Question Metadata Generation* convert the AMR graph into a set of intermediate AMR triples. Subjects and objects can be either *named entities* such as "`Benicio del Toro`" or *nominal entities* such as "`movie`" (referring a set of unknown movies). Named entities are linked to KB entities and nominal entities to KB classes. This information is passed to individual relation linking modules. Finally, the system generates a set of output triples with a scored ranked list of KB relations.

4 Question Metadata Generation

The components in *Question Metadata Generation*, process the question text and its AMR to produce the necessary metadata for relation linking components. The metadata include: (a) AMR triples, (b) KB entities and their types, and (c) answer type prediction.

AMR Graphs. As an input, `SLING` expects a richer semantic representation of the question generated by an AMR parser [13]. An AMR parse is a rooted, directed, acyclic graph expressing `"who is doing what to whom"` in a sentence or a question. Figure 1(b) shows a simplified version of an AMR graph for the question `Who is starring in Spanish movies produced by Benicio del Toro?`. Each node in the graph represents a concept, whereas edges represent relations between concepts that include ProbBank frames, nominal entities (types) and named entities. In this work, we rely on AMR graphs for the following reasons: (1) AMR detects named entities and maps them to predefined entity types (normalized) which forms the arguments of relations that have to be mapped to a KB, (2) AMR not only identifies relations in text but also normalises them using *PropBank* frames; (3) It reduces the ambiguity of natural language by converting relation phrases to their corresponding sense and (4) for questions, a special node, *amr-unknown*, is used to represent a placeholder for the answer to the question. Furthermore, the root node of the AMR graph,

[1] https://github.com/IBM/kbqa-relation-linking.

a.k.a the focus node, identifies the main focus of the question. Therefore, by using semantic parsing, we abstract out the syntactic variations and capture the meaning of the question in a more normalised manner.

AMR Graph to AMR Triples. DBpedia has only binary relations (two arguments). However, frames in AMR can have more than two arguments. For example, the `produce-01`[2] frame can have four core roles; `creator` (arg0), `creation` (arg1), `created from` (arg2), and `benefactive` (arg3) and other non-core roles such as time or location whereas on DBpedia there are only binary relations such as `dbo:producer`, `dbp:productionDate`, `dbo:basedOn`, or `dbo:location`. Despite the richer representation, this inherent mismatch between *n-ary* arguments of PropBank [7] frames and *binary* predicates in the KB poses a challenge. Therefore, it is necessary to generate AMR triples with a similar structure to KB triples (subject, predicate, and object) to facilitate their alignment. To resolve this issue, we use an approach that performs combinatorial expansion of all arguments of a frame to create binary relations and then prunes less probable combinations. More details of this process are presented in Sect. 5.2.

Entity/Type Linking and Answer Type Prediction. Once the AMR triples are derived, the next step is to link its subject and object to the KB. Subjects and objects from AMR can either be entities (Fig. 1: Bencio del Toro → `dbr:Bencio_del_Toro`) or classes (Fig. 1: movie → `dbo:Film`) in the KB. Entities are first linked to the KB using a regular entity linking tool that is based on BLINK [8] and DBpedia Lookup.

For classes, the mapping between AMR type system and DBpedia type system are generated semi-automatically. First, for each of 126 types from AMR type system (from AMR spec[3]), their instances are collected from AMR graphs and linked to KB entities. Then KB entity types are collected and they are ranked by frequency. Top 5 types are checked manually to map a KB type to each AMR type. This is a one time process that takes ∼2 h. This mapping can be performed against any type system (e.g., DBpedia, Wikidata) given a tool for entity linking is available. For the special node, `amr-unknown`, we map it to a KB-type by using an LSTM-based answer type prediction model. For instance, given a question such as "`Who is starring in Spanish movies produced by Benicio del Toro?`", it predicts `dbo:Actor` as the answer type.

5 Distantly Supervised Relation Linking

The question metadata such as AMR parse, AMR triples with entity and type information (from Sect. 4) are used as input to the four REL modules. Two of the modules that rely on distant supervision data are described below and the other two in the next Sect. 6.

Distantly supervised data is generally used in tasks where there is a lack of training data [11]. The lack of training data is also a significant challenge

[2] http://verbs.colorado.edu/propbank/framesets-english-aliases/produce.html.
[3] https://amr.isi.edu/doc/ne-types.html.

for REL tasks on KBQA datasets. Particularly, if we want to perform REL to DBpedia, we need training data for thousands of DBpedia relations. On the other hand, the KBQA datasets such as QALD and LC-QuAD 1.0 have 408 and 5000 questions covering a small subset of DBpedia relations. In order to address this issue, we collect training data using distant supervision, which eliminates the need for task-specific supervision for relation linking.

5.1 Distant Supervision Dataset

To train our REL models, for each relation, we require training examples (sentence, subject, object) mapped to its corresponding KB relation. For instance, as shown in Fig. 2 (Sentence: `Barack Obama was born in Honolulu, Hawaii`, subject: `Barack Obama`, object: `Honolulu, Hawaii`) mapped to (KB relation: `dbo:birthPlace`).

Corpus Pre-processing and Indexing: As shown in Fig. 2, we begin with the Wikipedia corpus, and perform co-reference resolution on each document. The corpus is then tokenized into sentences, and named entities are identified in each sentence to serve as `ElasticSearch` indices. We also store meta-data such as the document the sentence was extracted from and its position in the document. This meta-data is later used for selecting sentences.

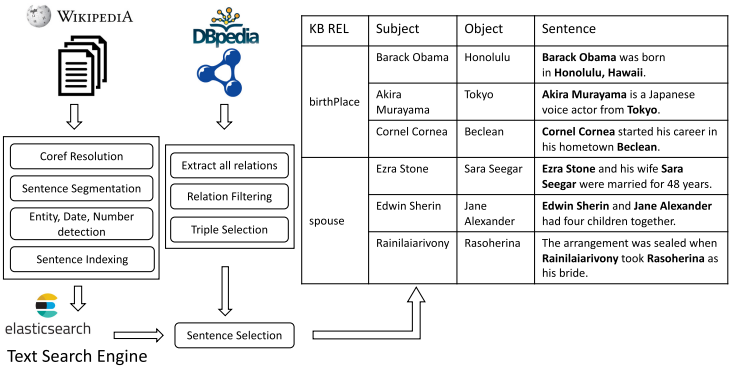

Fig. 2. Distant supervision data generation pipeline

Relation Selection: To address the issue of the large number of relations in KB, we select a manageable subset. DBpedia has a long tail of relations mainly due to uncommon Wikipedia infobox keys that are not widely used in queries. The number of examples that are generated by the distant supervision process depends on the number of triples containing the relation in the KB. While unsupervised modules use all relations in DBpedia, distantly supervised modules require some amount of examples to train the modules; thus the number of

relations used by them depends of their frequency of occurrence. The distance supervision process can generate more than 10 examples for $\sim$ 1.3K relations.

Selection of Examples: For each relation, we pick up to 1000 KB triples by ordering them by the sum of subject and object in-degrees. The assumption is that these entities are central and generally their corresponding Wikipedia articles contain more information. Then for each KB triple, we select a single example sentence, which is the first cooccurrence in entity's Wikipedia article. We choose sentences that satisfy the following: 1) subject and object co-occur, 2) have at least 4 tokens, 3) have at least 1 verb and 4) the entity surface forms do not overlap in the text (when one is a multi-word containing other). We observed that these basic heuristics increased the probability that the sentence contains a relation and filtered out accidental co-occurrences in titles, lists, etc.

5.2 Statistical AMR Predicate Alignment

This section presents a relation linking module that leverages the information present in the AMR semantic parses to generates alignments between *PropBank* predicates in AMR graphs and KB ontology relations. We describe below how these alignments are generated and then used to produce candidate relations.

Building PropBank Alignments. One challenge for creating these alignments is the inherent mismatch between frame-based representation and triple-based representation. In AMR graphs, a single frame captures a rich set of information using *n-ary* relations (e.g., who is doing what to whom, when, etc.) while triples in KBs capture simpler atomic facts using *binary* relations. For example, the frame `bear-02`[4], which is used to capture the event of giving birth to a child, has two core roles: `arg0` (mother), `arg1` (child) and several non-core roles including `location` (place of birth), `time` (time of birth) as shown in Fig. 3-A.

To address this mismatch in the number of arguments, we first decompose the AMR graph into a set of AMR triples. This is performed by creating binary relations between all entities participating in different roles of the frame using combinatorial expansion, as shown in Fig. 3-B. The generated binary relations are paths between the two nodes in the graph and follow the structure, <propbank-frame>.<subject-role>.<object-role>. Given a combination of two entities, for example, *Duka Tesla* (with the arg0 role) and *Nikola Tesla* (with the arg1 role), two AMR triples are generated, one with Dula Tesla as subject and Nikola Tesla as object and the other vice-versa as shown below:

```
Duka Tesla      bear02.arg0.arg1    Nikola Tesla
Nikola Tesla    bear02.arg1.arg0    Duka Tesla
```

Nevertheless, this process generates a large number of AMR triples that will not necessarily have their mapping relation in the KB. For example, in DBpedia, the place or the date that a mother gave a birth to a child

[4] http://verbs.colorado.edu/propbank/framesets-english-aliases/bear.html.

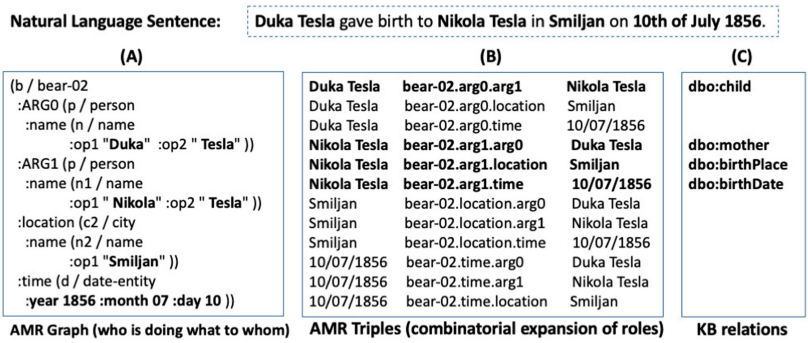

Fig. 3. converting AMR graphs to binary relations using combinatorial expansion

(bear02.arg0.location/time) is not represented as an attribute of the mother but only as attributes of the child and consequently there is no equivalent relation for those in the KB. This can be addressed by analysing how often we can align a given AMR triple to a KB triple. For example, out of 12 AMR triples generated (Fig. 3-B), only the four highlighted can be aligned with the existing KB triples in DBpedia.

Because KBs are generally multi-graphs and there are cases where two entities are connected with multiple relations in the KB. For example, if we assume Nikola Tesla was born and died in the same place, two entities (Nicola Tesla and Smijan) will be related both by `birthPlace` and `deathPlace` relations. In such cases with multiple candidates, we use lexical similarity between frame definition/aliases from PropBank (e.g., `bear`, `bear children`, `birth`, `give birth`) and DBpedia relation labels to disambiguate and select the most similar one.

Finally, to accommodate error propagation from both distant supervision dataset and AMR parsing, which could lead to noise in the alignments, we also use type constraints to further refine the alignments. The goal of this step is to induce type constraints for each role in a given frame. This is performed by collecting all entities participating in a given role in a frame (*e.g.*, bear-02.ARG0) and analyzing their types (including data types such as numerics and dates). Using this information, proxy domain and range constraints for AMR binary relations can be generated as in Fig. 4. These constraints are used to filter out any aligned DBpedia relation that does not match with the type constraints.

To summarize, for generating these alignments efficiently, we used the distant supervision dataset, defined as $D = \{(s_i, r_i, o_i, t_i), \dots\}$ where $\{s_i, r_i, o_i\}$ are the subject, relation, object of the KB triple and t_i is the corresponding sentence (see Fig. 2). We parse each t_i and generate an AMR graph a_i. Each a_i is then converted into a set of AMR triples $x_j = \{\grave{s}_j, p_j, \grave{o}_j\}$ where $x_j \in a_i$ and p_j is the AMR binary predicate, $\grave{s}_j$ and $\grave{o}_j$ are the subject and object from the AMR graph. Finally, we check for an AMR triple x_j where $\grave{s}_j = s_i$ and $\grave{o}_j = o_i$ and if found, one alignment between r_i and p_j is created.

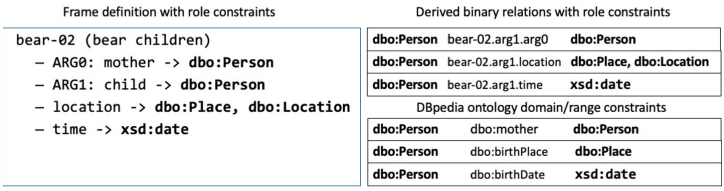

Fig. 4. Type constraints for frame roles

Finding Relation Candidates. Once the complete dataset is processed and alignments are filtered using type constraints, for each AMR binary predicate p_j we get a set of cumulative alignments $A(p_j) = \{(r_0, c_0), \ldots, (r_n, c_n)\}$ where each r is a KB relation and c is a alignment count. Using that, for each AMR binary predicate p_j, relation candidate scores are calculated using $relation\_score(p_j, r_n) = [c_n/max(c)] * [1/1 + log(inv\_pred\_count(r_n)]$ where $max(c)$ is highest count in $A(p_j)$ and $inv\_pred\_count(r_n)$ is the inverse predicate count, *i.e.*, number of distinct AMR predicates which r_n is aligned at least once.

5.3 Neural Model for Relation Linking

Statistical AMR mapping has the following drawbacks that can be addressed using a neural approach: (1) mapping generic frames such as `have-01` can be ambiguous. For example for: `‘‘Did Che Guevara have children?"` has the frame `have` that needs to be mapped to `dbo:child` (2) lexical gap where the same relation type can be expressed as different linguistic patterns. Therefore, we train a neural model for relation linking by exploiting the distant supervision dataset (Sect. 5.1). The neural model produces dense embedding vectors for input questions, which can learn to project the same relation type's different surface forms close in the latent space. In this section, we describe how to train the neural model on the distant supervision data (*training*) and how to make use of the model for question REL (*inference*).

Training Phase. Leveraging our distant supervision dataset, our training data is defined as $D = \{(x_0, r_0), \ldots, (x_N, r_N)\}$. Here $r_i \in R$ are the relation types, and $x_i = (t_i, s_i, o_i)$ are the relation instances consisting of a textual sentence t_i and the spans of the subject s_i and the object o_i. The set R represents the vocabulary of $K = |R|$ distinct relation types. We train a neural network M on D with the purpose to predict the correct relation type r_k given the instance x_k by minimize the cross-entropy loss regarding the conditional probability $p_M(\cdot|x_k)$ modeled by M, with respect to the true relation r_k.

In order to generate a vector representation of the relation instance x, we adopt the relation encoder inspired by [19]. This encoder is an adaptation of the original Transformer [23] architecture that encodes the given sentence while being aware of the subject and object. To achieve this entity-aware

encoding, we introduce four special tokens to mark the start and end positions of both entities in the sentence, `[SUBJ]`, `[\SUBJ]`, `[OBJ]` and `[\OBJ]` respectively. For instance, the second relation instance of `birthPlace` in Fig 2 is represented as "`[SUBJ] Akira Murayama [\SUBJ] is a Japanese voice actor from [OBJ] Tokyo [\OBJ]`". These new special tokens are randomly initialized and fine-tuned during training, whereas all the other tokens are initialized using the pre-trained BERT-BASE embeddings [3]. We concatenate the vectors of the final-layer hidden states of the start entity markers of subject and object entities, feed them into a fully connected layer to get the finally embedding vector for the relation instance x.

Finally, to estimate $p_M(r_k|x_k)$, we add a further classification layer with the output size K followed by a softmax function.

Inference Phase. There are several challenges to address when applying the trained neural relation linking model M to deal with question relationship linking. In particular, how to mark the missing entities from the question, which consists of two cases: (1) the missing entity is the answer; (2) the missing entity is an intermediate entity when the question requires multiple hops to reach the answer. We exploit the *AMR graph to AMR triples* feature described in Sect. 4 to handle these challenges.

- *Intermediate entities:* Consider the question in Fig. 1 and its generated metadata. The question requires to first find some Spanish movie entities having the `dbo:producer` relation with `Benicio del Toro`, e.g.., `7 días en La Habana`, then identifies another relation `dbo:star` from the movie entity. Since the movie name is missing in the question text, when predicting its relationship to `Benicio del Toro`, we take the surface form of `arg2` for `star-01.arg2.arg1`, i.e., the word "`movies`" as the object. In this way, we generate the following input relation instance to the neural model M, `Who is starring in Spanish [OBJ] movies [\OBJ] produced by [SUBJ] Benicio del Toro [\SUBJ] ?`.
- *Unknown (answer) entities:* Consider the same question as above. The predicate `star-01.arg2.arg1` has no explicit text for the `arg1` since the `amr-type` is `unknown`, which refers to the answer. In this case, we mark the question word "Who" for the `arg2`. Therefore the following format for the relation instance is generated for our neural model: "`[OBJ] Who [\OBJ] is starring in Spanish [SUBJ] movies [\SUBJ] produced by Benicio del Toro?`".

Finally, with the aforementioned treatments, for each relation instance a ranked list of relation types in DBpedia is generated and sorted by their probability scores produced by our neural model.

6 Unsupervised Relation Linking and Score Aggregation

In the previous section, we have described the 2 distantly supervised modules. In this section, we describe the remaining 2 modules and aggregation of scores to get the final ranked list of KB relations.

6.1 Unsupervised Modules

Lexical Similarity. To derive the score of a relation with respect to an AMR triple, we compute its lexical similarity to the question text and AMR predicate. For each relation candidate, like `dbo:deathPlace`, we consider its label as a word sequence `death place`. We concatenate each question, e.g., `Who was married to Lincoln`, with the AMR predicate of the triple (e.g. `marry` from `marry-01`) to get the other word sequence. We compute the lexical similarity between the two word sequences by first calculating a word-by-word cosine similarity based on word2vec embeddings. If there are m words in one word sequence, and n words in the other, this produces $m \times n$ similarity scores. This is max-pooled to produce a single score as output.

Knowledge Base Connections. In KBQA, the entities from the questions are identified and linked to KB first. Therefore, the task of relation linking also assumes the existence of such linked KB entities and entity types as described in the *Question Metadata Generation* step. Hence, candidate relations that also connect these detected entities can be scored higher, following previous works [16]. For example, given the question "`Who created Family Guy?`", to predict the relation in this question, we score all relations connected to the KB entity (`dbr:Family_Guy`) as the object and a subject of KB type `dbo:Person` or any subclass of it (which is predicted by answer type prediction). We then apply a soft constraint to focus more on the relations that are within this set.[5].

6.2 Score Aggregation

The scores from each module are normalized using min-max normalization. The final score of a relation is the arithmetic sum of its normalized score from each module, and a ranked list of relations is obtained for the AMR triple. This process is repeated for every AMR triple extracted from the question.

7 Evaluation

In this section, we detail our experimental setup and evaluate our approach against the state-of-the-art relation linking approaches for KBQA. We replicate the experimental setup proposed in Falcon [16] in terms of the same datasets and metrics used for a fair comparison, as described below.

[5] Ideally, we can do the hard filtering with the relation connections. However as REL is a component of a whole KBQA pipeline. To mitigate potential error propagation from entity linking, most works adopt a soft approach [6,16].

Table 1. KBQA datasets statistics

Dataset	Questions	Avg constraints
QALD-7	215	1.5
QALD-9	408	1.5
LC-QuAD 1.0	5000	1.7

Table 2. Relation Linking systems comparison

	QALD-7			LC-QuAD 1.0			QALD-9		
System	P	R	F1	P	R	F1	P	R	F1
SIBKB	0.29	0.31	0.30	0.13	0.15	0.14	–	–	–
ReMatch	0.31	0.34	0.33	0.15	0.17	0.16	–	–	–
EARL	0.27	0.28	0.27	0.17	0.21	0.18	–	–	–
Falcon	**0.58**	0.61	0.59	**0.42**	0.44	0.43	0.31	0.34	0.32
SLING	0.57	**0.76**	**0.65**	0.41	**0.58**	**0.48**	**0.50**	**0.64**	**0.56**

7.1 Experimental Setup

We used three KBQA datasets; QALD-7 [22], QALD-9 [21] and LC-QuAD 1.0 [20]. All the datasets comprise of question text, their corresponding SPARQL queries, and answers from DBpedia. Similar to [16], we use the question text and the relations in the SPARQL queries for evaluation[6]. Table 1 shows the number of questions and the average triple constraints in SPARQL queries for each of the datasets. QALD-9 is an evolved version of QALD-7 extending the number of questions from 215 to 408.

We compare SLING against four existing REL approaches for KBQA: (1) SIBKB [18], (2) ReMatch [12], (3) EARL [4], and (4) Falcon [16]. Falcon [16] is the state-of-the-art approach evaluated on QALD-7 and LC-QuAD 1.0 datasets. We use standard metrics such as precision, recall, and F-measure for evaluation and comparisons. The precision measures the capability of a REL system to predict the exact number of expected relations in a given question and the recall measures the capability of a system to cover all the expected relations.

7.2 Results

Table 2 shows the precision, recall, and F-measure of SLING in comparison to state-of-the-art approaches. The results show that our approach consistently achieves a better F1 score than the existing approaches and is robust across datasets, i.e. the results on QALD-7 and QALD-9 are respectively similar compared to those obtained by Falcon[7]. Moreover, SLING provides a remarkably higher recall than the other competing systems.

[6] We exclude `rdf:type`, and `rdfs:label` to follow same setting in [16].

[7] Falcon numbers on QALD-7 and LC-QuAD 1.0 are taken from their paper.

Ablation Study: In order to understand the contribution of each module in the `SLING` framework, we perform an ablation study by removing the corresponding module from the overall system and comparing its performance. These results reported in Table 3 indicates that every module contributes to the overall performance of the system results. Particularly, removing the statistical AMR mapping approach from the system has the biggest drop in performance. The AMR mapping component provides the strongest contribution of the modules, with the system performance dropping considerably without its usage. AMR provides predicates that are already a strong signal for identifying the relations in a sentence. Moreover, AMR parsers normalize syntactic variations across sentences which have the same meaning. Finally, AMR provides type information about the subject and object of a relation, even when they are unknown. This enables domain and range-derived features to constrain the predicted relation candidates.

Table 3. Ablation study on QALD-7 dataset

	P	R	F1
SLING	0.57	0.76	0.65
w/o AMR Mapping	0.45	0.57	0.51
w/o Neural Relation Linking	0.52	0.66	0.58
w/o Word Embeddings	0.53	0.68	0.59
w/o KB Analysis	0.46	0.61	0.53

Table 4. Relation linking performance with machine generated vs human annotated AMR on a subset of QALD-9 dataset

	P	R	F1
w/ machine generated AMR	0.53	0.76	0.62
w/ human annotated AMR	0.57	0.77	0.66

For relations that are implicit in text, the Neural Relation Linking bridges the lexical gap to map them. For instance, considering the first example in Table 6, the relation type `dbo:country` is implicit in the question, but the neural model is able to identify it nevertheless. Furthermore, the Neural Relation Linking is able to handle questions having more than one relation, where different relations can be predicted given the same question text, but different spans of entities, as described in Sect. 5.3. This insight confirms that the distant supervision technique can be helpful in covering different language phrases to identify and link relations to a KB, especially in setting such as QALD, where only a small training set is provided.

Table 5. Relation linking performance with our entity linking implementation vs annotations from [14] on LC-QuAD dataset.

	P	R	F1
w/ our entity linker implementation	0.41	0.58	0.48
w/ entity annotations from [14]	0.46	0.62	0.53

Based on our analysis we find that the Word Embeddings module is particularly useful when the relation is explicitly mentioned in the question, like `'Who is the mayor of Paris?'`, with the relation being `dbo:mayor`. It provides high-precision estimates about the relations in the question.

Automatic extraction of DBpedia triples from Wikipedia Infoboxes (when mappings are not available) introduces redundant and noisy (`dbp:`) relations. For instance, there are relations such as `dbo:birthPlace`, `dbp:birthPlace`, `dbp:birthLocation` and `dbp:placeOfBirth` that are semantically equivalent and cannot be lexically distinguished based on their labels. In such scenarios KB analysis allows the system to choose the correct relation by considering only the ones connected to the entities of interest. For the question `'What is the birth place of Frank Sinatra?'`, without the KB analysis we find `dbo:birthPlace` and `dbp:placeOfBirth` as the top ranked relations. KB analysis scores `dbp:placeOfBirth` higher because of its association with the entity `dbr:Frank_Sinatra` (Table 5).

Impact of AMR Parser: To understand how the quality of AMR affects the results, we have manually annotated a subset of QALD-9 questions (ids 250 to 408) and the results are presented in Table 4. It shows that human annotated AMRs provide an improvement of 4 points in F1. The state-of-the-art AMR parser [13] has a smatch score of 90% when tested on a subset of QALD-9 dataset.

Impact of Entity Linking: To understand the effect of entity linking, we have performed a similar experiment using entity annotations provided by [14] for LC-QuAD 1.0. We have tested the entity linker implementation we used with QALD-9; it has an F1 of 0.75.

7.3 Qualitative Analysis and Discussion

Table 6 shows five example questions with their gold standard relations compared to what `SLING` predicts for each question. `SLING` was able to find the correct set of relations for the first three questions and partially solves the rest. In the first question, the main challenge is to decompose the question into three triples with correct subject/object combinations. Leveraging AMR allows `SLING` accurately determine the correct triple decomposition including directionality and the number of triples. Once decomposed, all relation linking modules provide strong signals in this example. The second and third questions are lexically

Table 6. Example queries with gold and predicted relations.

ID	Question	Gold standard triple patterns	Predicted relations
1	Who is starring in Spanish movies produced by Benicio del Toro?	?film dbo:starring ?actor . ?film dbo:country res:Spain . ?film dbo:producer res:Benicio_del_Toro	dbo:starring dbo:country dbo:producer
2	Who developed Skype?	res:Skype dbo:developer ?company	dbo:developer
3	Who developed Slack?	?company dbo:product res:Slack	dbo:product
4	Give me the grandchildren of Bruce Lee.	res:Bruce_Lee dbo:child ?child ?child ?dbp:child ?granchild	dbo:child
5	Which organizations were founded in 1950?	{ ?org dbo:formationYear ?date } UNION { ?org dbo:foundingYear ?date } UNION { ?org dbp:foundation ?date } UNION { ?org dbp:formation ?date }	dbo:foundingYear

very similar but their representations in KB are different. The fact that `SLING` creates triples with directionality into account and perform KB analysis allowing it to pick correct relation in each case. The fourth question is challenging because it requires to reason that grand children are children of children. AMR represent this using a single triple. Furthermore, the relation used in each constraint is different (`dbo:child` vs `dbp:child`). `SLING` gets a set of candidates such as `dbo:child`, `dbp:children`, `dbp:grandChilden` and picks `dbo:child` as it is connected to `res:Bruce_Lee`. Nevertheless, it does not decompose this question into two triples. Similarly, some gold standard questions have the `UNION` construct with logically equivalent relations in KB. However, as the relation appears only once in the text, `SLING` only aims to predict one relation.

8 Conclusions and Future Work

In this paper, we presented `SLING`, a framework for relation linking that leverages semantic parsing with AMR and distant supervision. `SLING` is a combination of multiple modules that capture complementary signals both from the AMR representation as well as natural language text. Experimental results show that `SLING` outperforms state-of-the-art approaches on three KBQA datasets; QALD-7, QALD-9, and LC-QuAD 1.0. Furthermore, our ablation study shows that leveraging AMR and the use of distant supervision contributes to outperform the state-of-the-art techniques. As a part of our future work, we are planning to convert all the components as feature generators for an end to end neural approach. Furthermore, we intend to investigate the use of transformer-based architectures for encoding both AMR graphs and the question text for relation linking.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
2. Chakraborty, N., Lukovnikov, D., Maheshwari, G., Trivedi, P., Lehmann, J., Fischer, A.: Introduction to neural network based approaches for question answering over knowledge graphs. CoRR abs/1907.09361 (2019)
3. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT 2019, pp. 4171–4186 (2019)
4. Dubey, M., Banerjee, D., Chaudhuri, D., Lehmann, J.: EARL: joint entity and relation linking for question answering over knowledge graphs. In: Vrandečić, D., et al. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 108–126. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_7
5. Höffner, K., Walter, S., Marx, E., Usbeck, R., Lehmann, J., Ngomo, A.N.: Survey on challenges of question answering in the semantic web. Semant. Web **8**(6), 895–920 (2017)
6. Hu, S., Zou, L., Yu, J.X., Wang, H., Zhao, D.: Answering natural language questions by subgraph matching over knowledge graphs. IEEE Trans. Knowl. Data Eng. **30**(5), 824–837 (2018)
7. Kingsbury, P., Palmer, M.: PropBank: the next level of treebank. In: Proceedings of Treebanks and lexical Theories, vol. 3. Citeseer (2003)
8. Wu, L., Petroni, F., Josifoski, M., Riedel, S., Zettlemoyer, L.: Zero-shot entity linking with dense entity retrieval. arXiv:1911.03814 (2019)
9. Lehmann, J., et al.: DBpedia-a large-scale, multilingual knowledge base extracted from Wikipedia. Semant. Web **6**(2), 167–195 (2015)
10. Lukovnikov, D., Fischer, A., Lehmann, J.: Pretrained transformers for simple question answering over knowledge graphs. In: Ghidini, C., et al. (eds.) ISWC 2019. LNCS, vol. 11778, pp. 470–486. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30793-6_27
11. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: ACL 2009, pp. 1003–1011 (2009)
12. Mulang, I.O., Singh, K., Orlandi, F.: Matching natural language relations to knowledge graph properties for question answering. In: SEMANTiCS 2017, pp. 89–96 (2017)
13. Naseem, T., Shah, A., Wan, H., Florian, R., Roukos, S., Ballesteros, M.: Rewarding smatch: Transition-based AMR parsing with reinforcement learning. arXiv preprint arXiv:1905.13370 (2019)
14. Pan, J.Z., Zhang, M., Singh, K., Harmelen, F., Gu, J., Zhang, Z.: Entity enabled relation linking. In: Ghidini, C., et al. (eds.) ISWC 2019. LNCS, vol. 11778, pp. 523–538. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30793-6_30
15. Rossiello, G., Gliozzo, A., Farrell, R., Fauceglia, N.R., Glass, M.R.: Learning relational representations by analogy using hierarchical siamese networks. In: NAACL-HLT (1), pp. 3235–3245. Association for Computational Linguistics (2019)
16. Sakor, A., et al.: Old is gold: linguistic driven approach for entity and relation linking of short text. In: NAACL: HLT 2019, pp. 2336–2346 (2019)

17. Singh, K., Both, A., Sethupat, A., Shekarpour, S.: Frankenstein: a platform enabling reuse of question answering components. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 624–638. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_40
18. Singh, K., et al.: Capturing knowledge in semantically-typed relational patterns to enhance relation linking. In: K-CAP 2017, pp. 1–8 (2017)
19. Soares, L.B., FitzGerald, N., Ling, J., Kwiatkowski, T.: Matching the blanks: distributional similarity for relation learning. In: ACL 2019, pp. 2895–2905 (2019)
20. Trivedi, P., Maheshwari, G., Dubey, M., Lehmann, J.: LC-QuAD: a corpus for complex question answering over knowledge graphs. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10588, pp. 210–218. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68204-4_22
21. Usbeck, R., Gusmita, R.H., Ngomo, A.N., Saleem, M.: 9th challenge on question answering over linked data (QALD-9) (invited paper). In: Semdeep/NLIWoD@ISWC. CEUR Workshop Proceedings, vol. 2241, pp. 58–64. CEUR-WS.org (2018)
22. Usbeck, R., Ngomo, A.-C.N., Haarmann, B., Krithara, A., Röder, M., Napolitano, G.: 7th open challenge on question answering over linked data (QALD-7). In: Dragoni, M., Solanki, M., Blomqvist, E. (eds.) SemWebEval 2017. CCIS, vol. 769, pp. 59–69. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69146-6_6
23. Vaswani, A., et al.: Attention is all you need. In: NIPS, pp. 5998–6008 (2017)
24. Yu, M., Yin, W., Hasan, K.S., dos Santos, C.N., Xiang, B., Zhou, B.: Improved neural relation detection for knowledge base question answering. In: ACL 2017, pp. 571–581 (2017)