Entity Linking for KGQA Using AMR Graphs

Nadine Steinmetz$^{[0000-0003-3601-7579]\star}$

Technische Universität Ilmenau, Ilmenau, Germany
`nadine.steinmetz@tu-ilmenau.de`
Bergische Universität Wuppertal, Wuppertal, Germany
`steinmetz@uni-wuppertal.de`

Abstract. Entity linking is an essential part of analytical systems for question answering on knowledge graphs (KGQA). The mentioned entity has to be spotted in the text and linked to the correct resource in the knowledge graph (KG). With this paper, we present our approach on entity linking using the abstract meaning representation (AMR) of the question to spot the surface forms of entities. We re-trained AMR models with automatically generated training data. Based on these models, we extract surface forms and map them to an entity dictionary of the desired KG. For the disambiguation process, we evaluated different options and configurations on QALD-9 and LC-QuaD 2.0. The results of the best performing configurations outperform existing entity linking approaches.

Keywords: Entity Linking · AMR · Data Augmentation.

1 Introduction

The correct identification of named entities in a natural language (NL) question is a key challenge for question answering on knowledge graphs (KGQA) systems. In terms of analytical approaches, the challenge of entity linking includes two aspects: the correct recognition of the surface form of the named entity within the text, as well as the correct disambiguation of possibly ambiguous phrases.

With this paper, we present our approach on entity linking in the context of KGQA [17]. Our proposed approach on entity linking is bipartite: a trained approach for the identification of the surface form and an analytical approach for the disambiguation and linking process.

We utilize the abstract meaning representation (AMR) of the question to structure the input question in a graph and identify separate parts of the question. Several libraries provide pre-trained models for the transformation. These models are using language models for the analysis of the syntax of the question, and training data, e.g. by the Linguistic Data Consortium (LDC), to transform a NL sentence/question to an AMR graph. We identified several flaws of the pre-trained models. We therefore generated augmented training data and re-trained several AMR models to be evaluated. The models are available via Zenodo: `https://zenodo.org/record/7442882`.

* Work was done while at TU Ilmenau

The second part of our approach is the actual entity linking process. We utilize dictionaries containing several context information for the named entities within the respective KG. We rank possible entity candidates and choose the candidate with the highest score as the most relevant for the input question and surface form.

For the evaluation of our approach, we generated test data from actual KGQA test datasets – QALD-9 [20] and LC-QuaD 2.0 [5]. We compared our approach to existing approaches and their results on those test datasets. The results show that our approach outperforms other competing approaches on both datasets.

We automatically generated training data to re-train our AMR model and eliminate the identified flaws of the existing model. Our evaluation results show that with this augmentation and re-training, our new model improves the entity linking by at least 5 % in recall and precision. Overall, we processed 28 different configurations for each test dataset to identify the best performing configuration from three different categories: AMR model, context information, calculation of ranking score.

The remainder of the paper is structured as follows: Related work is described in Section 2. We introduce our data augmentation process and the parameters of the re-training of the AMR models in Section 4. The entity linking step including the description of the dictionary and the ranking process is depicted in Section 5. We evaluated our approach on two different data sets based on Wikidata and DBpedia. The results are shown and discussed in Section 6. We summarize our approach and discuss future work in Section 7.

2 Related Work

In 2008, Milne et al. presented their approach on *wikification* – linking mentions in text to Wikipedia pages [14]. Their approach is completely based on machine learning, trained on Wikipedia articles. Both steps, the identification of phrases within the text to be linked to Wikipedia pages as well as the disambiguation of ambiguous phrases are trained on their training dataset. The disambiguation process utilizes unambiguous phrases within the context of the ambiguous phrase to help find the correct meaning and Wikipedia page.

TagMe also links text to Wikipedia pages and has been published in 2010 by Ferragina et al. [6]. Their approach aims at very short texts – which corresponds to entity linking in the context of QA. Their approach also takes into account the context of the text and the authors claim to propose a method that computes ranking scores for all meanings (resp. candidates) very fast. The disambiguation includes the calculation of the *relatedness* of Wikipedia pages in the context.

One of the first approaches of entity linking on DBpedia was DBpedia Spotlight [13] in 2011. Mendes et al. presented an approach consisting of several sub steps: spotting, candidate mapping, disambiguation and linking. For the spotting, a string matching algorithm based on Aho-Corasick is used. Afterwards, candidates are selected using the DBpedia Lexicalization dataset[1]. This stage

[1] Unfortunately, the Lexicalization dataset is not available anymore.

of the process already pre-ranks the candidates. The subsequent disambiguation makes use of a vector space model where each candidate is a vector in the space. The disambiguation then uses context information of the input text/question to rank the candidates in the vector space.

Also in 2011, AIDA has been presented by Yosef et al. [21]. AIDA uses for the spotting of entities in a text the Stanford NER tagger[2]. The disambiguation process utilizes a graph-based approach. The nodes in the graph are mentions and entities. There are two types of edges: edges between entities and edges between a mention and an entity. For a text, such a graph is created using the mentions (surface forms) of the text and adding the respective entities as predefined. Then the entity-entity edges are added. The goal of their approach is to reduce the graph to a high density graph where each mention is linked to only one entity.

Falcon 2.0 originally has been introduced as entity and relation linking tool over Wikidata by Sakor et al. [16]. Meanwhile the authors provide an API and the annotation of text also includes links to DBpedia entities and relations. The spotting of the surface forms in a text utilizes tokenization and n-gram tiling to retrieve combined tokens as surface forms. To identify the correct candidates, the approach tries to find triples in the KG that involves pairs of relations and entities from the candidate lists. Each match within the KG increases the individual score of the entity and relation candidates.

As stated by Bender et al. [1], large language models (LLM), such as BERT [3], are claimed to understand the meaning of NL while they actually are trained to predict text based on linguistic form. These LLMs might not be able to understand NL, but they can help to analyze and transform NL to other formal languages. Therefore, those LLMs are also suitable for the task of entity linking.

BLINK is an entity linking tool that utilizes bi-encoder to spot entity mentions and a cross-encoder to disambiguate and link the mentions to Wikipedia pages. The authors utilized a fine-tuned BERT architecture for that purpose. Later, the BLINK architecture has been enhanced especially for entity linking on questions (ELQ) [12]. The major enhancement of the ELQ architecture is a performance boost by processing multiple entity mentions at once and the eliminated necessity of mention boundaries with the input. BLINK is integrated as built-in wikification service in several libraries for the generation of AMR graphs, such as amrlib.

The approach of OpenTapioca has been published in 2019 [2].The author proposes a light weight model for the purpose of entity linking on Wikidata which can be used to be run or trained to keep a linking system up to date as the Wikidata KG changes. The approach takes into account the local compatibility of pairs of mentions and entities as well as semantic similarity of entities occurring as candidates in the same context of a text. These metrics are computed based on the actual KG and used for training the model.

In 2021, Jiang et al. presented their neuro-symbolic approach which includes interpretable rules and neural learning [8]. Amongst other, the authors evaluated

[2] `http://nlp.stanford.edu/software/CRF-NER.shtml`

their approach on QALD-9, but their test dataset contains 174 target entities (compared to 125 in our test dataset). Probably, they test dataset includes also categories and subjects in addition to only named entities. Therefore, we did not include their results in our comparison of evaluation results.

In terms of data augmentation and improving AMR parsing, Lee et al. presented several experiments to improve AMR parsing on different data sets [10]. AMR graphs for the QALD-9 test and train datasets were created manually by three annotators over a year and they achieved a SMATCH score of 89.3. This score is only 1.6 % higher than the score we achieved with our automatic data augmentation process (c.f. Section 4).

As discussed in Section 4.1, AMR are very suitable as intermediate representation for KGQA applications due to their graph nature. Besides our own approach [17], there are already other existing approaches on KGQA utilizing AMR in the transformation pipeline from NL to SPARQL, as e.g. [15] and [9]. Both latter approaches use BLINK for the entity linking process.

3 Approach Overview

Our presented approach consists of two main sub steps. The first step is independent from the desired KG and identifies the surface forms of potential named entities within the input question. In the second step, the surface form is mapped to the respective KG to identify and disambiguate the specific named entity.

The first step includes the generation of the AMR graph and the extraction of the relevant parts of the surface forms of the named entities. Named entities are referenced using a *name* node in the graph.

With this approach, we are able to outperform existing competing approaches for entity linking based on Wikidata and DBpedia. As shown in Section 6, our data augmentation is able to improve the AMR model. Based on the re-trained models, we apply the entity mapping and ranking. The results for the overall process are very promising compared to competing systems. Our approach is described more in detail in the next sections.

For the AMR generation, we evaluated several libraries and identified several problems with the pre-trained model. Therefore, we re-trained the model for the AMR generation. The motivation and data augmentation process are described in Section 4. The entity linking process after extracting the surface forms from the AMR graph is described in Section 5.

4 Data Augmentation & Training

There are already existing libraries and models for the generation of AMR graphs. The most prominent libraries are amrlib[3] and JAMR[4]. Both libraries provide pre-trained models. While JAMR is older in general, the pre-trained

[3] `https://github.com/bjascob/amrlib`
[4] `https://github.com/jflanigan/jamr`

model naturally has been trained on older training data provided by the Linguistic Data Consortium (LDC)[5]. In contrast, amrlib was updated last in June 2022[6]. The developer provides a pre-trained model based on BART Large [11] and utilizing AMR 3.0 training data by the LDC[7]. We utilize the amrlib library for our approach. Section 4.1 gives a short introduction on AMR graphs. Section 4.2 describes flaws of the pre-trained model and our motivation to re-train the model. Section 4.3 depicts the data augmentation process to generate more training data and eliminate the identified flaws of the pre-trained model.

4.1 Abstract Meaning Representation

AMR graphs are directed, labeled, acyclic graphs. Each graph corresponds to one sentence[8]. An important aspect of AMR graphs is the negligence of the specific syntax of natural language. Sentences with similar meaning are aimed to obtain the same AMR graph even if they are phrased differently. This characteristic is essential for KGQA as different natural language questions can lead to the same formal query.

The AMR specification[9] explains the roles (used as edge property) and node labels to describe NL in a graph. For QA, the `amr-unknown` node label is essential as it represents the unknown fact a question asks for. Consider the following AMR graph for the question *Who is the mayor of Berlin?*:

```
(z0 / mayor
     :domain (z1 / amr-unknown)
     :location (z2 / city
                    :name (z3 / name
                               :op1 "Berlin")))
```

The node `mayor` has two child nodes: the one for Berlin and the one for an unknown resource. This means, the query constructed to answer this question must find something which is connected to Berlin through a relationship called `mayor`. This syntax is very close to the way, how facts are represented in KGs. In this way, AMR graphs are eligible as intermediate representation between an NL question and the formal query[10].

Named entities are represented using a `name` node with a parent node that describes the name node in a categorical way (namely as `city` in our example). The child nodes of the `name` node contain the parts of the surface form. Thus, the identification of named entities is straightforward using AMRs: finding `name` nodes in the graph and collecting the labels of their child nodes.

However, the correctness of the graphs depends on the sample data utilized to train the model. For our purpose of entity linking, we identified some problems

[5] The newest model is trained on 2016 LDC training data.
[6] as of December 2022
[7] `https://catalog.ldc.upenn.edu/LDC2020T02`
[8] c.f. `https://amr.isi.edu/index.html`
[9] `http://www.isi.edu/~ulf/amr/help/amr-guidelines.pdf`
[10] as shown in [9] and [17]

with the existing models trained on the LDC AMR 3.0 dataset. The next section describes those issues in detail.

4.2 LDC Training Data & its flaws

Since 2013, the LDC released datasets containing sample AMR graphs. The latest dataset – AMR 3.0 – has been released in January 2020 containing 59,255 sample AMR graphs from 13 categories. The dataset contains manually generated AMR graphs for sentences "from broadcast conversations, newswire, weblogs, web discussion forums, fiction and web text"[11]. The main problem of this training data is the case sensitivity of all sentences. This means, the casing of the words and surface forms (for named entities) is always correct. But for user-generated content – especially in QA scenarios – correct casing cannot be expected and this is a critical issue for the generation of correct AMR graphs[12]. Consider the question *What type of film is on the nose?*. The question refers to the movie *On the Nose* released in 2001 starring Dan Aykroyd and Robbie Coltrane. Please observe the incorrect casing of the surface form of the movie. Unfortunately, the model trained on the LDC AMR 3.0 dataset produces the following AMR graph for the question with incorrect casing:

```
(o / on-the-nose
      :domain (f / film
            :mod (t / type
                  :mod (a / amr-unknown))))
```

In addition, we identified another problem with the pre-trained model regarding the entity linking task. Surface forms of named entities containing single quotes are also problematic in some cases. For the question *What is the significance of artists of The Beatles' Story?* the pre-trained model only identifies *The Beatles* as a named entity of type *story*.

Therefore, we decided to re-train the model regarding these issues. The next section describes the process of generating the training data automatically.

4.3 Generation of augmented Training Data

We utilized the train splits of the KGQA benchmark datasets LC-QuaD 2.0 (based on Wikidata) [5] and QALD-9 (based on DBpedia) [20] for our approach. Both datasets are QA datasets containing NL questions, the respective SPARQL queries, and the answers from the underlying KG.

For our purpose, we created entity linking datasets by extracting the named entities from the SPARQL queries. As we identified two different issues with the pre-trained model, we also followed different approaches to automatically generate training data. We describe the generation process briefly for both approaches in the following paragraphs.

[11] `https://catalog.ldc.upenn.edu/LDC2020T02`

[12] cf. the training dataset of the SMART task challenge 2022: `https://smart-task.github.io/2022/`

Casing Problem For each question of the training dataset, we check if the main label(s) of the respective named entities are contained in the question in correct casing. If so, we utilize the pre-trained model to generate the AMR graph and check if all components of the main labels of the contained entities are correctly contained in the graph as child nodes of the `name` nodes. If so, the question and the phrases of the child nodes in the AMR graph are converted to lowercase again and the graph is added as training data. We generate more similar training records using named entities of the same type as the required named entity in the original question. For instance, *On the Nose* is of `rdf:type dbo:Film`. We retrieve more instances of type `Film` and repeat the data augmentation process with the main labels of those entities. In this way, we created over 100,000 additional training records regarding the casing issue.

Surface Forms with Single Quotes Again, for each question of the training datasets, we retrieve the referenced named entities and their respective ontology types. If the entity is referenced in the question using the main label, we generate similar questions utilizing the main labels of named entities of the same type containing single quotes. We enclose the surface form of the referenced entity with double quotes and generate the AMR graph. In this way, we generate the question *Give me all actors starring in movies directed by and starring "Lil' JJ".* from the original of question of the QALD9 training dataset: *Give me all actors starring in movies directed by and starring William Shatner.*. If all components of the surface form are contained as child nodes in the name node, we remove the double quotes in the question and add the graph to the training data. In this way, we generated another 2,500 training graphs.

Utilizing the generated training data, we re-trained three AMR models in addition to the already pre-trained model. In the remainder of the paper, the different models are referred to as following:

- LDC – model only trained on LDC AMR 3.0
- LDC+LC – model trained on LDC AMR 3.0 + augmented data for lower cased questions
- LDC+QU – model trained on LDC AMR 3.0 + augmented data with entities containing quotes in the label
- LDC+QU+LC – model trained on LDC AMR 3.0 + all augmented data

We utilized BART Large [11] as well as the PEGASUS language model[13] and evaluated different configurations on both options.

For the training process, the training data is split into train, dev and test splits (0.7 : 0.1 : 0.2) and the correctness of the predicted graphs are evaluated using a SMATCH score[14]. We achieved the best results utilizing BART Large with a batch size of 16 and 32 epochs for all models. The SMATCH scores for this configuration are 0.819 (LDC+QU) and 0.877 (LDC+LC and LDC+QU+LC)

[13] `https://huggingface.co/docs/transformers/model_doc/pegasus`
[14] `https://amr.isi.edu/smatch-13.pdf`

respectively. Hence, the best training process achieves a SMATCH score 4 percent higher than the score achieved only using AMR 3.0 as training data[15]. We evaluated all four models for our further entity linking process. The models are available as download[16] [17].

5 Entity Linking

The second part of our entity linking process is the actual mapping of extracted surface forms to the desired KG and choosing the most probable candidate in case of ambiguous phrases. Section 5.1 describes the extraction of the surface forms from the AMR graph in detail. Section 5.2 gives an insight into our mapping dictionary and the characteristics of the named entity candidates we take into account. The ranking process is described in Section 5.3.

5.1 Label Extraction from the AMR graph

As already mentioned, named entities are referenced using a `name` node an associated child nodes in the AMR graph. For our sample question *what type of film is on the nose?* the child nodes are connected to the `name` node using the edge roles `:op1`, `:op2`, and `:op3`. We collect the node labels of the child nodes and order them according to their edge roles.

But, we noticed some remarkable results of the AMR generation. In some cases, the AMR graph contains phrases that are not contained in the input question. Consider the question *What file format is the environment of Grand Theft Auto III?*. Our re-trained model (and also the pre-trained model) generates an AMR graph that contains a name node for the phrase *Grand Watch Auto III*. Apparently, this results from the language model (in our case BART) which replaces the probable unknown node label with a known label.

Therefore, we generate all k-grams of the input question, where `k` is the number of token in the surface form extracted from the AMR graph. We calculate the similarity of all k-grams with the surface form using Levenshtein distance. The k-gram with the lowest distance – above the threshold of a minimum similarity of 70 % – is chosen as the respective surface form.

5.2 Mapping to underlying KB

For the dictionary of entities within the underlying KG, we utilize various information. The labels are collected from main labels and alternative labels. As we use RDF/OWL KGs – Wikidata and DBpedia, the main labels are

[15] The SMATCH score for BART Large is stated as 0.837 trained on LDC AMR 3.0

[16] `https://doi.org/10.5281/zenodo.7442882`

[17] instructions on how to use the models with amrlib can be found here: `https://github.com/bjascob/amrlib-models`

collected using `rdfs:label` information. Alternative labels are collected differently, depending on the KG. For DBpedia, we use the main labels of redirects and disambiguation resources. Wikidata provides alternative labels using `skos:alternativeLabel` as property. With this collection, we receive a wide range of labels but also increase the ambiguity of many labels in our dictionary. Therefore, we added a score for each label that corresponds to the distance of the label to the main label of the entity. In general, we calculate the Levenshtein distance, but also take into account if the label is an abbreviation of the main label or a synonym or a commonly used substring, such as a family name of a person. The calculation of the score is described more in detail in [18].

In addition, we use the indegree of the entity when considered a node in the network of entities. For DBpedia, we utilize the incoming Wikipedia page links. Wikidata provides a property `sitelinks` which corresponds to the indegree of page links.

And lastly, we add some context information to the entities in the dictionary. For DBpedia, we collect all labels of classes the entity is an instance of (in terms of `rdf:type`). These classes include umbel, yago and DBpedia ontology classes. For Wikidata, we utilize the property `instance of` (P31) to collect descriptive information for the entities.

Overall, the dictionary for the lookup contains the following information:

- URI – identifier of the named entity
- lowercase – the lowercased label
- score – distance of the label to the main label of the entity
- lowercase_stemmed – the stemmed version of the lowercase label
- indegree – the page link indegree resp. sitelinks
- types – context information (`rdf:type` resp. `instance of`)

Our dictionary for DBpedia entities contains over 19 million entries and the dictionary for Wikidata contains over 96 million entries.

The mapping to the entity dictionary is processed in three levels where the respective next level is only accessed if the current level does not provide results:

1. unstemmed equal mapping on the lowercase column
2. stemmed equal mapping on the lowercase_stemmed column
3. fuzzy search on the lowercase column using similarity function

In general, we retrieve the top 10 results using the similarity score for the fuzzy search. But also the first two steps of equal mappings can result in more than one entity candidate. Therefore, a ranking of the results and subsequent choice of the most relevant candidate is necessary. The ranking process is described in the next section.

5.3 Ranking

The ranking process is necessary in case multiple entity candidates are retrieved during the mapping process. The entity candidates possess three different features: a score – either the distance to the main label or in case of fuzzy search the similarity to the surface form, between 0.0 and 1.0, indegree, and type information.

For the ranking, we take into account the label-based score (in the remainder of the paper referred to as label score s_l) and calculate two additional scores for each entity candidate:

- score for the indegree – referred to as indegree score s_i, and
- a context-based score based on the types of the candidate and the context information of the question, referred to as context score s_c.

All scores are normalized to a value between 0.0 and 1.0.

The context score s_c requires context information from the input question. In QA scenarios, this context information is often very little, but the AMR graph sometimes provides additional information not contained in the question. For instance, the parent node of the `name` node for "Berlin" – from the example above – has the node label `city`. This is information not contained in the question itself. Additional context information can be collected using the node labels of all nodes of the AMR graph except for operator nodes, such as `amr-unknown`. The context for ranking the entity candidates for *Berlin* would thereby constitute of *city*, and *mayor*[18].

We evaluated both options of context creation: only the label of the parent node or the labels of all nodes in the graph. We want to emphasize that we use the parent node (in terms of a categorical type of the named entity) as context information for the disambiguation process and do not restrict the entity candidates to this type when mapping to the KG. The KG might not contain that information or is erroneous. The context score is then calculated based on how much of the context information from the question is contained in the types of the entity candidates.

For the final ranking, we make use of these three scores s_i, s_l, and s_c, and calculate a ranking score s_r. For the evaluation, we tested various combinations of weights for the different scores. The results are shown in Table 2 and discussed in Section 6.

Finally, the entity candidate with the highest ranking score s_r is chosen.

6 Evaluation

We evaluated our approach on two datasets and against six other entity linking approaches. We want to emphasize the following aspects of our evaluation:

- influence of the data augmentation process on the model
- influence of different options of context information
- evaluation of different configurations regarding the different scores for label distance, indegree and context matching

We introduce the datasets in Section 6.1. The experiments and the results are described and discussed in Section 6.2.

[18] which would not be helpful for that disambiguation case

6.1 Datasets

For the evaluation of our approach, we utilize two different datasets. We want to attach importance on datasets that are appropriate for the actual QA process. That means, the dataset should contain the SPARQL queries that are required to answer the questions based on the respective KG. Therefore, we decided to utilize QALD-9 [20] and LC-QuaD 2.0 [5]. A pure entity linking dataset, such as NILK[19], is therefore not under consideration for our evaluation. More insights regarding contained types etc. on QALD-9 and the previous version of LC-QuaD 2.0 can be found in [19].

6.2 Experiments

In a first step, we evaluated different configurations for our approach. There are three aspects of the configuration of our approach:

- choice of AMR model
- creation of context information – only the parent node of the `name` node or labels of all nodes in the graph
- calculation of the ranking score – consisting of the context score s_c, the label score s_l and the indegree score s_i

Taking into account the three different scores, we calculated seven different ranking scores s_r:

- all scores unweighted: $s_r = (s_i + s_l + s_c) \ / \ 3$
- one score double weighted respectively (makes 3 additional configurations), e.g. $s_r = ((2 * s_i) + s_c + s_l)/4$
- combination of two scores double weighted (makes another 3 additional configurations), e.g. $s_r = ((2 * s_i) + (2 * s_c) + s_l)/5$

Overall, we generated 14 different configurations per dataset and model.

6.3 Results

In the following, we present and discuss the results for our specific focal points of the evaluation separately.

Choice of AMR model We generated the results for all 14 configurations for each model and dataset. Table 1 shows the best results respectively. For both datasets, we can observe that our data augmentation process was able to improve the results. All results on re-trained models using our augmented data are increased compared to the results on the basic AMR model. Remarkably, the overall best results are achieved using the LDC+LC model (as expected LDC+QU+LC would achieve best results). Apparently, the model creates less correct graphs when trained on LDC+QU+LC. The amount of augmented data

[19] `https://zenodo.org/record/6607514`

might play a role here – over 100,000 graphs for LC versus 2,500 for QU. Also, the structure of the sentences repeat, as we utilized the same training dataset to generate the augmented data. Therefore, the same sentences with different entities appear multiple times in the training data. This might result in conflicting graphs and we will include this aspect in future work.

Table 1. Evaluation results for the respective best configuration utilizing the different AMR models and on both datasets.

	AMR model											
	LDC			LDC+LC			LDC+QU			LDC+QU+LC		
Dataset	R	P	F1	R	P	F1	R	P	F1	R	P	F1
QALD-9	85.7	82.8	84.2	**89.1**	**85.7**	**87.4**	86.7	82.8	84.7	87.7	82.9	85.2
LC-QuaD 2.0	59.7	64.6	62.1	**65.3**	**70.3**	**67.7**	63.8	68.0	65.8	64.8	69.7	67.2

Table 2 shows the results for all configurations utilizing the best performing AMR model – LDC+LC – for each dataset. We will discuss our results regarding the aspects of context creation and ranking score in the next paragraphs.

Creation of context information For the context aspect, the results are clear. For both datasets, recall and precision are higher when the context information only consists of the node label of the parent node of the `name` node. As the parent node is a descriptive node of the `name` node, this is the most specific information for the disambiguation of the `name` node. We assume, the rest of the AMR graph to be too distractive as it describes also other name nodes and of course the actual result of the question.

Ranking score The results for the calculation of the ranking score are not that clear at least for the different datasets. On QALD-9 the best results are achieved having all scores unweighted or doubled weight for indegree and context score – conf 1 and conf 7 respectively. Apparently, the weighting of the label score decreases the overall result. But, the differences of recall and precision are too marginal for a definite conclusion.

For LC-QuaD the differences between score calculations are again not too significant. And only one combination achieves the best results: a doubled weighting of the label score and the indegree score. For this dataset, the context information does not seem to be too relevant. A tentative conclusion could be the accentuation of the indegree score. As the questions in the dataset often provide only minimal context, the popularity of the mentioned named entities might be of importance. A high indegree score emphasizes the most popular of the entity candidates.

With the experiments as described above, we were able to identify the configurations that achieve the best results on the utilized datasets. We also compared our results – naturally for the best configuration – with competing approaches.

Table 2. Results of our approach for 14 different configurations on QALD-9 and LC-QuaD 2.0 using the AMR model trained on LDC 3.0 and AUG LC

training data	QALD-9						LC-QuaD 2.0					
context	parent			all nodes			parent			all nodes		
	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1
conf 1: s_l, s_i, s_c	**89.1**	**85.8**	**87.4**	87.7	84.8	86.2	64.4	69.1	66.7	65.0	70.0	67.4
conf 2: s_l, s_i, $2s_c$	87.2	84.3	85.7	86.2	83.3	84.7	63.6	68.3	65.9	64.7	69.4	67.0
conf 3: $2s_l$, s_i, s_c	88.2	84.7	86,4	86.2	82.8	84.5	64.6	69.2	66.8	65.2	70.0	67.5
conf 4: s_l, $2s_i$, s_c	88.2	85.2	86.6	87.7	84.8	86.2	64.7	69.7	67.1	64.6	69.6	67.0
conf 5: $2s_l$, $2s_i$, s_c	88.2	85.2	86.6	87.7	84.8	86.2	**65.3**	**70.3**	**67.7**	65.1	70.1	67.5
conf 6: $2s_l$, s_i, $2s_c$	87.2	83.8	85.5	85.8	81.2	83.4	64.0	68.7	66.3	64.8	69.5	67.1
conf 7: s_l, $2s_i$, $2s_c$	**89.1**	**85.8**	**87.4**	87.7	84.8	86.2	64.0	68.7	66.3	64.5	69.5	66.9

Comparison with competing approaches We compared the results of our approach to six competing systems: AIDA, DBpedia Spotlight, Falcon 2.0 API, OpenTapioca, TagMe, and BLINK. For the LC-QuaD 2.0 dataset, we used the results published by the developers ([16] for Falcon 2.0 API) and by Diomedi et al. for the results of AIDA, DBpedia Spotlight, OpenTapioca, and TagMe [4].

We also compared our results on QALD-9 to the results of DBpedia Spotlight, TagMe, and Falcon 2.0 API. We retrieved the results of them by using the provided APIs[20]. BLINK is available as built-in option in the amrlib for the AMR generation. Links to named entities referenced to the English Wikipedia respectively DBpedia within the AMR graph are referenced with a `:wiki` tag. BLINK is also available as standalone version without the use of AMR. We evaluated both options. OpenTapioca only provides results for Wikidata and AIDA does not provide an API feasible for evaluation tasks[21].

All results of the competing systems compared to our best results are shown in Table 3. The Falcon 2.0 API does not provide configuration parameters. For DBpedia Spotlight, we achieved the best results using a confidence score of 0.6. The response of the TagMe API contains a `rho` parameter which corresponds to a confidence score. Without threshold, the recall for TagMe is 73 % and the precision as low as 30 %. In [7] the authors used a threshold of 0.1 for the `rho` parameter (achieving recall= 73.1 %, precision= 39.0, F_1-score= 50.9 %), but we achieved the best result in terms of F_1-score using `rho`=0.2, c.f. Table 3.

Obviously, our approach outperforms the other approaches. For QALD-9, we achieve a more than 10 % higher recall than Falcon 2.0 API and even over 17 % higher precision. For LC-QuaD 2.0, the difference to the best competing system is as high as 6 % in terms of recall compared to TagMe, but our precision is 20 % higher than by Falcon 2.0 API. Overall, we could show that our approach

[20] Falcon: https://labs.tib.eu/falcon/falcon2/api-use,
Spotlight: `https://www.dbpedia-spotlight.org/api`,
TagMe: `https://sobigdata.d4science.org/web/tagme/tagme-help`

[21] The authors do not wish to use the JSON web service for evaluation comparison and it also responses with HTTP 404 as of December 10th 2022.

Table 3. Results of our approach compared to other approaches

	QALD-9			LC-QuaD 2.0		
	R	P	F_1	R	P	F_1
DBpedia Spotlight	73.1	70.1	71.6	52.5	23.3	30.8
TagMe	70.2	52.2	59.9	59.4	29.5	37.4
AIDA	n.a.	n.a.	n.a.	30.5	38.5	33.1
BLINK (amrlib built-in)	64.4	61.0	62.5	n.a.	n.a.	n.a.
BLINK (standalone)	79.0	74.6	76.7	n.a.	n.a.	n.a.
OpenTapioca	n.a.	n.a.	n.a.	42.0	29.0	35.0
Falcon 2.0 API	78.0	68.1	72.7	56.0	50.0	53.0
Our approach	**89.1**	**85.8**	**87.4**	**65.3**	**70.3**	**67.7**

is efficient and achieves very good results. We will discuss questions where our approach fails further in the next section.

6.4 Discussion

We identified several reasons for failures and wrong linking processes of our approach:

- wrong AMR generation,
- missing label for surface form in entity dictionary,
- disambiguation errors, and
- specific characteristics of the SPARQL query.

There are two different aspects of wrong AMR generation: wrong structure / identification of named entities, and the modification of question phrases in the AMR graph. In the first case, the AMR model is not able to identify the named entity in the question correctly and does not provide a `name` node or only parts of the actual surface form of the entity in the `name` node.

A second problem with the AMR generation is the modification of question parts within the AMR. As already described in Section 5.1, we compare the label constructed from the child nodes of the `name` nodes with n-grams of the question. Consider the question *When did Dracula's creator die?*. The AMR model modifies *Dracula* to *drago*. The levenshtein distance between *drago* and *Dracula* is 5 – when case sensitivity is considered – and too high for our predefined threshold.

The lexical gap is still a problem when it comes to QA scenarios as named entities can be referenced with multiple surface forms. For instance, the question *Which subsidiary of TUI Travel serves both Glasgow and Dublin?* asks for the airports of Dublin and Glasgow which both are represented by own named entities in the underlying KG. But, the airports are only referenced by mentioning the names of their location.

Disambiguation errors are also a major challenge especially when only little context is given. For instance, the QALD-9 dataset contains the question

What are the names of the Teenage Mutant Ninja Turtles?. Our approach disambiguates the surface form *Teenage Mutant Ninja Turtles* to the entity of the movie `dbr:Teenage_Mutant_Ninja_Turtles_(2014_film)`. But the query asks for the more general resource of the original series. We often see a similar behavior of our approach when the question asks for a movie or a movie character. Mostly, the most popular entity candidate (as per our best performing configuration) is chosen which is often not the entity for the original movie or character.

Lastly, another challenge is the comprehension of the underlying KG. In some cases the detected named entities are not relevant for the specific SPARQL query dependent on the KG. Or the SPARQL query requires more entities not even mentioned in the question to retrieve the answer for the question. Consider the task *Give me all taikonauts.* The required SPARQL query for that question based on DBpedia requires the ontology class `dbo:Astronaut` and references to the named entities of P.R. China. Our approach links *taikonauts* to `dbr:Astronaut`. Clearly, this challenge cannot be solved only using entity dictionaries.

7 Summary

We presented our two-fold approach that makes use of AMR graphs to analyze the syntax of a question. In the second step, we follow an analytical approach to avoid the out-of-vocabulary problem, but also to be able to apply our algorithm to any KG. With our data augmentation process, we were able to improve the AMR model by re-training it using automatically generated training data.

We provide an exhaustive evaluation taking into account context options and weights for the entity candidate ranking. Our results show that the categorical description of a name node should be preferred over using all information of the question as context for the disambiguation. Unfortunately, the evaluation of weights on the ranking scores does not show clear results to draw conclusions. The two datasets are using different KGs and therefore different characteristics of the KGs could be the cause. Future work would include evaluations on more datasets based on DBpedia and Wikidata.

We also discussed failures of our approach. Some of the issues might be eliminated by additional training data. For instance, the model could be trained to not split surface forms in case of consecutive words beginning with an upper case. Another problem is the modification of the phrases of the question in the AMR graph. We need to examine additional training and parsing parameters to prevent this modification.

Overall, our presented approach shows very promising results as it outperforms other existing entity linking systems. Our future work includes the further improvement of our approach as briefly discussed above.

Acknowledgement

The author wants to thank Khaoula Benmaarouf and Kanchan Shivashankar for their help in preparing some of the data utilized for this paper.

References

1. Bender, E.M., Koller, A.: Climbing towards NLU: On meaning, form, and understanding in the age of data. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 5185–5198. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.463, `https://aclanthology.org/2020.acl-main.463`
2. Delpeuch, A.: Opentapioca: Lightweight entity linking for wikidata. CoRR **abs/1904.09131** (2019), `http://arxiv.org/abs/1904.09131`
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2018). https://doi.org/10.48550/ARXIV.1810.04805, `https://arxiv.org/abs/1810.04805`
4. Diomedi, D., Hogan, A.: Entity linking and filling for question answering over knowledge graphs. In: Natural Language Interfaces for the Web of Data (NLIWOD) Workshop (2022)
5. Dubey, M., Banerjee, D., Abdelkawi, A., Lehmann, J.: Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In: Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I.F., Hogan, A., Song, J., Lefrançois, M., Gandon, F. (eds.) The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11779, pp. 69–78. Springer (2019). https://doi.org/10.1007/978-3-030-30796-7_5, `https://doi.org/10.1007/978-3-030-30796-7\_5`
6. Ferragina, P., Scaiella, U.: Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management. p. 1625–1628. CIKM '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1871437.1871689, `https://doi.org/10.1145/1871437.1871689`
7. Ferragina, P., Scaiella, U.: Fast and accurate annotation of short texts with wikipedia pages. IEEE Software **29**(1), 70–75 (2012). https://doi.org/10.1109/MS.2011.122
8. Jiang, H., Gurajada, S., Lu, Q., Neelam, S., Popa, L., Sen, P., Li, Y., Gray, A.: LNN-EL: A neuro-symbolic approach to short-text entity linking. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 775–787. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.acl-long.64, `https://aclanthology.org/2021.acl-long.64`
9. Kapanipathi, P., Abdelaziz, I., Ravishankar, S., Roukos, S., Gray, A.G., Astudillo, R.F., Chang, M., Cornelio, C., Dana, S., Fokoue, A., Garg, D., Gliozzo, A., Gurajada, S., Karanam, H., Khan, N., Khandelwal, D., Lee, Y., Li, Y., Luus, F.P.S., Makondo, N., Mihindukulasooriya, N., Naseem, T., Neelam, S., Popa, L., Reddy, R.G., Riegel, R., Rossiello, G., Sharma, U., Bhargav, G.P.S., Yu, M.: Leveraging abstract meaning representation for knowledge base question answering. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021. Findings of ACL, vol. ACL/IJCNLP 2021, pp. 3884–3894. Association for Computational Linguistics (2021). https://doi.org/10.18653/v1/2021.findings-acl.339, `https://doi.org/10.18653/v1/2021.findings-acl.339`

10. Lee, Y.S., Astudillo, R., Thanh Lam, H., Naseem, T., Florian, R., Roukos, S.: Maximum Bayes Smatch ensemble distillation for AMR parsing. In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 5379–5392. Association for Computational Linguistics, Seattle, United States (Jul 2022). https://doi.org/10.18653/v1/2022.naacl-main.393, `https://aclanthology.org/2022.naacl-main.393`
11. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension (2019). https://doi.org/10.48550/ARXIV.1910.13461, `https://arxiv.org/abs/1910.13461`
12. Li, B.Z., Min, S., Iyer, S., Mehdad, Y., Yih, W.t.: Efficient one-pass end-to-end entity linking for questions. In: EMNLP (2020)
13. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: Shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems. p. 1–8. I-Semantics '11, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2063518.2063519, `https://doi.org/10.1145/2063518.2063519`
14. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management. p. 509–518. CIKM '08, Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1458082.1458150, `https://doi.org/10.1145/1458082.1458150`
15. Neelam, S., Sharma, U., Karanam, H., Ikbal, S., Kapanipathi, P., Abdelaziz, I., Mihindukulasooriya, N., Lee, Y., Srivastava, S.K., Pendus, C., Dana, S., Garg, D., Fokoue, A., Bhargav, G.P.S., Khandelwal, D., Ravishankar, S., Gurajada, S., Chang, M., Uceda-Sosa, R., Roukos, S., Gray, A.G., Lima, G., Riegel, R., Luus, F.P.S., Subramaniam, L.V.: SYGMA: system for generalizable modular question answering overknowledge bases. CoRR **abs/2109.13430** (2021), `https://arxiv.org/abs/2109.13430`
16. Sakor, A., Singh, K., Patel, A., Vidal, M.E.: Falcon 2.0: An entity and relation linking tool over wikidata. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. p. 3141–3148. CIKM '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3340531.3412777, `https://doi.org/10.1145/3340531.3412777`
17. Shivashankar, K., Benmaarouf, K., Steinmetz, N.: From graph to graph: Amr to sparql. In: Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference (ESWC 2022), Hersonissos, Greece, May 29th, 2022. CEUR Workshop Proceedings, vol. 3196. CEUR-WS.org (2022), `http://ceur-ws.org/Vol-3196`
18. Steinmetz, N.: Context-aware semantic analysis of video metadata. Ph.D. thesis, University of Potsdam (2014), `http://opus.kobv.de/ubp/volltexte/2014/7055/`
19. Steinmetz, N., Sattler, K.U.: What is in the kgqa benchmark datasets? survey on challenges in datasets for question answering on knowledge graphs. Journal on Data Semantics **10**(3-4), 241–265 (2021). https://doi.org/10.1007/s13740-021-00128-9
20. Usbeck, R., Gusmita, R.H., Ngomo, A.N., Saleem, M.: 9th challenge on question answering over linked data. In: Choi, K., Anke, L.E., Declerck, T., Gromann, D., Kim, J., Ngomo, A.N., Saleem, M., Usbeck, R. (eds.) Joint proceedings of the 4th

Workshop on Semantic Deep Learning (SemDeep-4) and NLIWoD4: Natural Language Interfaces for the Web of Data and 9th Question Answering over Linked Data challenge (QALD-9) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, California, United States of America, October 8th - 9th, 2018. CEUR Workshop Proceedings, vol. 2241, pp. 58–64. CEUR-WS.org (2018), `http://ceur-ws.org/Vol-2241/paper-06.pdf`

21. Yosef, M.A., Hoffart, J., Bordino, I., Spaniol, M., Weikum, G.: Aida: An online tool for accurate disambiguation of named entities in text and tables. Proc. VLDB Endow. **4**(12), 1450–1453 (aug 2011). https://doi.org/10.14778/3402755.3402793, `https://doi.org/10.14778/3402755.3402793`