Introduction to Neural Network based Approaches for Question Answering over Knowledge Graphs

Nilesh Chakraborty[*], Denis Lukovnikov[*],
Gaurav Maheshwari[†], Priyansh Trivedi[†],
Jens Lehmann[‡], Asja Fischer[§]

Article Type:

Overview

Abstract

Question answering has emerged as an intuitive way of querying structured data sources, and has attracted significant advancements over the years. In this article, we provide an overview over these recent advancements, focusing on neural network based question answering systems over knowledge graphs. We introduce readers to the challenges in the tasks, current paradigms of approaches, discuss notable advancements, and outline the emerging trends in the field. Through this article, we aim to provide newcomers to the field with a suitable entry point, and ease their process of making informed decisions while creating their own QA system.

[*]Joint First Author; Smart Data Analytics Group, University of Bonn

[†]Joint First Author; Enterprise Information Systems, Fraunhofer IAIS

[‡]Smart Data Analytics Group, University of Bonn & Enterprise Information Systems Department, Fraunhofer IAIS

[§]University of Bochum, Germany

1 Introduction

Advancements in semantic web technologies and automated information processing systems have enabled the creation of a large amount of structured information. Often, these structures follow well defined formal syntax and semantics, enabling machine readability and in some cases interoperability across different sources and interfaces. A prime example of such a structure is a special kind of graph based data model referred to as *knowledge graph* (KG). Due to their expressive nature, and the corresponding ability to store and retrieve very nuanced information, accessing the knowledge stored in KGs is facilitated through the use of formal query languages with a well-defined syntax, such as SQL, SPARQL, GraphQL etc. However, the use of formal queries to access these knowledge graph pose difficulties for non-expert users as they require the user to understand the syntax of the formal query language, as well as the underlying structure of entities and their relationships. As outlined by Hirschman and Gaizauskas (2001), a KG question answering (KGQA) system therefore aims to provide the users with an interface to ask questions in natural language, using their own terminology, to which they receive a concise answer generated by querying the KG. Such systems have been integrated in popular web search engines like Google Search[1] and Bing[2] as well as in conversational assistants including Google Assistant[3], Siri[4], and Alexa. Their applicability in domain specific scenarios has also been demonstrated, for example in the Facebook Graph Search and IBM Watson[5].

Due to their wide appeal, a variety of approaches for KGQA have been proposed. Traditional approaches (Unger et al., 2015; Berant et al., 2013; Reddy et al., 2014, etc.) typically involve a composition of template extraction, feature engineering, and traditional semantic parsing based methods. More recently, increasingly many neural network based approaches have been shown to be effective for the KGQA task as well. These approaches range from simple neural embedding based models (Bordes et al., 2014), over attention based recurrent models (Cheng et al., 2019), to memory-augmented neural controller architectures (Liang

[1] http://www.google.com/
[2] http://www.bing.com
[3] https://assistant.google.com/
[4] https://www.apple.com/siri/
[5] https://www.ibm.com/watson

et al., 2017; Neelakantan et al., 2017; Yang et al., 2017).

This article provides an introduction to neural network based methods for KGQA. In Section 2, we provide the necessary background related to KGQA, introducing the terminology used in the community and the major tasks solved by KGQA systems. In Section 3, we provide a brief overview of datasets over which various KGQA systems train and benchmark their performance. Section 4 introduces readers to the current paradigms of neural network based KGQA approaches, and provides an in-depth overview of each of these paradigms including methods for training and inference, as well as challenges specific to each paradigm, and common means to overcome them. We conclude our discussion by listing a few emerging trends and potential future directions of research in this area in Section 5. We assume familiarity of readers with the basic deep learning methods and terminology and refer readers to Goodfellow et al. (2016) and the survey article of Goldberg (2016) for good overviews on deep learning and applications of neural network models in natural language processing, respectively.

2 Background

This section provides a brief introduction of the concepts necessary for an in-depth understanding of the field of KGQA, including a formal definition of KGs, a description of formal query languages, and a definition of the KGQA task and associated subtasks.

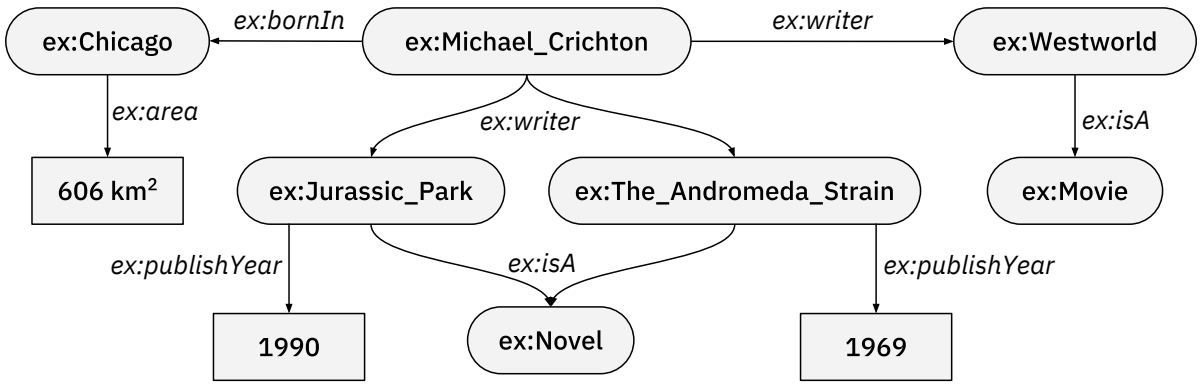

Figure 1: An example KG which will serve as a running example throughout this article.

2.1 Knowledge Graphs

A KG is a formal representation of facts pertaining to a particular domain (including the general domain). It consists of *entities* denoting the subjects of interest in the domain, and *relations*[6] denoting the interactions between these entities. For instance, ex:Chicago and ex:Michael_Crichton can be entities corresponding to the real world city of Chicago, and the famous author Micheal Crichton respectively. Further, ex:bornIn can be a relation between the two denoting that Micheal Crichton was born in Chicago. Instead of linking two entities, a relation can also link an entity to an data value of the entity (such as a date, a number, a string etc.) which is referred to as a *literal*. The entity ex:Chicago for example might be connected via a relation ex:area to a numerical literal with the value of $606km^2$ describing the size of city. Figure 1 depicts a snippet of an example KG representing knowledge about the writer Michael Crichton, where rounded boxes denote entities, squared boxes literals, and labelled edges denote relations.

Formally, let $\mathcal{E} = \{e_1 \dots e_{n_e}\}$ be the set of entities, $\mathcal{L}$ be the set of all literal values, and $\mathcal{P} = \{p_1 \dots p_{n_p}\}$ be the set of relations connecting two entities, or an entity with a literal. We define a *triple* $t \in \mathcal{E} \times \mathcal{P} \times (\mathcal{E} \cup \mathcal{L})$ as a statement comprising of a subject entity, a relation, and an object entity or literal, describing a fact. For instance, one of the facts in the KG depicted in Fig. 1 can be written as $\langle$ex:Michael_Crichton, ex:bornIn, ex:Chicago$\rangle$. Then, a KG $\mathcal{K}$ is a subset of $\mathcal{E} \times \mathcal{P} \times (\mathcal{E} \cup \mathcal{L})$ of all possible triples representing facts that are assumed to hold.

Remark: A common relation rdfs:label is used to link any resource (entity or relation) with a literal signifying a human readable name of the resource. For instance, $\langle$ex:Michael_Crichton, rdfs:label, "Michael Crichton"@ en$\rangle$. We refer to these literals as the *surface form* of the corresponding resource, in the rest of this article.

2.2 Formal Query Languages

A primary mechanism to retrieve and manipulate data stored in a KG is provided by formal query languages like SPARQL, λ-DCS (Liang, 2013), or FunQL (Kate and Mooney, 2006).

[6]The words *predicate* or *property* are often used interchangeably with relations.

These languages have a well defined formal grammar and structure, and allow for complex fact retrieval involving logical operands (like disjunction, conjunction, and negation), aggregation functions (like grouping or counting), filtering based on conditions, and other ranking mechanisms.

SPARQL, a recursive acronym for "SPARQL Protocol and RDF Query Language", is one of the most commonly used query languages for KGs and is supported by many publicly available KGs like DBpedia (Lehmann et al., 2015) and Freebase (Bollacker et al., 2008). For an in-depth understanding of the syntax and semantics of SPARQL, we refer interested readers to the W3C Technical Report[7]. The central part of a SPARQL query is a graph, composed of resources from the KG (i.e. entities and relations) and variables to which multiple KG resources can be mapped. As a running example that we will use throughout the rest of the paper, we consider the KG presented in Figure 1 to be the source KG over which we intend to answer the natural language query (NLQ) "*What is the birthplace of Westworld's writer?*" The corresponding SPARQL query is illustrated in Figure 2.

```
What is the birthplace of Westworld's writer?
```

```
SELECT ?uri WHERE {
    ?x ex:writer ex:Westworld.
    ?x ex:bornIn ?uri
}
```

Figure 2: An example of (a) a (factual) NLQ and (b) a SPARQL query corresponding to the NLQ, which when executed over the KG presented in Fig. 1 would return ex:Chicago as the result.

Another popular formal query representation language is lambda calculus (Barendregt, 1984), a formal language for defining computable functions originally devised by Church (1936) and used for his study of the *Entscheidungsproblem*(Ackermann and Hilbert, 1928). The example NLQ "What is the birthplace of Westworld's writer" can be expressed as follows in lambda calculus:

$$\lambda x.\exists y.\mathsf{bornIn}(y, x) \wedge \mathsf{writer}(x, \mathsf{Westworld}) \ . \tag{1}$$

SPARQL and lambda calculus are both rather verbose and may result in longer and syntactically complex expressions than necessary for their use in question answering. λ-DCS (Liang,

[7] https://www.w3.org/TR/rdf-sparql-query/#introduction

2013) and FunQL (Kate and Mooney, 2006) both provide a more concise query representation than SPARQL or lambda calculus by avoiding variables and making quantifiers from lambda calculus implicit. In λ-DCS, our example NLQ can be simply expressed as $\mathsf{bornIn.writer.Westworld}$. The equivalent FunQL expression is $\mathsf{bornIn(writer(Westworld))}$. For purposes of further exposition, we will now introduce the FunQL language in more details, adhering to the definitions provided in Cheng et al. (2019). The FunQL language operates over sets. The language consists of the following components: (1) entities symbols (e.g. $\mathsf{Barack\_Obama}$) whose denotation is the singleton set containing that entity (e.g. $[\![\mathsf{Barack\_Obama}]\!] = \{\mathsf{Barack\_Obama}\}$), (2) relation functions (e.g. $\mathsf{spouse()}$), which return the set of entities that are reached by following the relation from its argument (e.g. $[\![\mathsf{spouse(Barack\_Obama)}]\!] = \{\mathsf{Michelle\_Obama}\}$) and (3) additional set functions, including (a) $\mathsf{count()}$, which returns the cardinality of a set (e.g. $\mathsf{count(spouse(Barack\_Obama))} = 1$, (b) various filtering functions, which filter a set using a relation and a filtering value (e.g. $[\![\mathsf{filter}_<\mathsf{(spouse(Barack\_Obama), birthdate, 1990)}]\!] = \{\mathsf{Michelle\_Obama}\}$), as well as (c) $\mathsf{argmax()}$, $\mathsf{argmin()}$ and (d) set intersection ($\mathsf{and()}$) and union ($\mathsf{or()}$).

2.3 Question Answering Over Knowledge Graphs

Using the concepts introduced above, we now define the task of KGQA. Let $\mathcal{K}$ be a KG and let q be an NLQ then we define the set of all possible answers $\mathcal{A}$ as the union of (i) the power set $\mathcal{P}(\mathcal{E} \cup \mathcal{L})$ of entities $\mathcal{E}$ and literals $\mathcal{L}$ in $\mathcal{K}$, (ii) the set of the numerical results of all possible aggregation functions $f : \mathcal{P}(\mathcal{E} \cup \mathcal{L}) \mapsto \mathbb{R}$ (such as SUM or COUNT), and (iii) the set $\{\mathsf{True}, \mathsf{False}\}$ of possible boolean outcomes, which is needed for yes/no questions. The task of KGQA then is defined as returning the correct answer $a \in \mathcal{A}$, for a given question q.

KGQA systems often use a *semantic parser* to accomplish their task. Semantic parsing is the task of translating an NLQ q into an executable meaning representation $f \in \mathcal{F}$ of q (see also Kamath and Das (2019) for a recent survey on semantic parsing). $\mathcal{F}$ is the set of all formal queries (see Section 2.2) that can be generated by combining entities and relations from $\mathcal{K}$ as well as arithmetic/logical aggregation functions available in the formal query

[7] Set of all subsets of a set of elements.

language. The correct logical form f must satisfy the following conditions: (1) the execution of f on $\mathcal{K}$ (which we denote by $f(\mathcal{K})$) yields the correct answer a, as implied by the NLQ and (2) f accurately captures the meaning of the question q. The second condition is important because multiple logical forms could yield the expected results a but not all of them *mean* the same thing as the question q. We call such logical forms that satisfy the first constraint but not the second *spurious* logical forms. For an example of spurious logical forms for KGQA lets consider the following NLQ "What books did Michael Crichton write before 1991?". The queries $\mathsf{filter}_{<}(\mathsf{and}(\mathsf{isA}(\mathsf{Novel}), \mathsf{writtenBy}(\mathsf{Michael\_Crichton})), \mathsf{publishingYear}, 1991)$ and $\mathsf{and}(\mathsf{isA}(\mathsf{novel}), \mathsf{writtenBy}(\mathsf{Michael\_Crichton}))$ both execute to the same set of books because Michael Crichton didn't write books after 1991. However, the second query is incorrect since it does not convey the full meaning of the question.

A note on terminology: The *natural language question* (which we abbreviate to *NLQ*), is also referred to as *question* or *utterance*. The *meaning representation* is also referred to as *logical forms* or *formal queries*. The *execution results* or *answers* for a formal query are also referred to as *denotations*.

2.4 KGQA Subtasks

Regardless of the specific semantic parsing approach taken, the construction of logical forms for KGQA requires taking several decisions regarding their structure and content. Here, we briefly highlight those different tasks the semantic parser will have to perform.

1. **Entity Linking**: Entity linking in the context of KGQA is the task of deciding which KG entity is referred to by (a certain phrase in) the NLQ q. In the case of our example, the entity linker must identify $\mathsf{ex{:}Westworld}$ as the entity being referred to be "Westworld" in the NLQ. The large number (often millions) of entities forms a core challenge in entity linking with large-scale KGs. A particular phrase (e.g. "Westworld"), if considered without context, can refer to several different entities (Westworld: the movie, the series, the band or the album). This phenomenon is referred to as *polysemy*. It is essential to take the context of the phrase into account to correctly *disambiguate*

the phrase to the correct entity. The large number of entities makes it practically unfeasible to create fully annotated training examples to learn lexical mappings for all entities. As a consequence, statistical entity linking systems need to generalize well to unseen entities. Most modern KGQA systems externalize the task of entity linking by employing a standalone entity linking system like DBpedia Spotlight (Mendes et al., 2011)for DBpedia or S-Mart (Yang and Chang, 2016) for Freebase.

2. **Identifying Relations**: It is essential to determine which relation must be used in a certain part of the logical form. Similarly to entity linking, we need to learn to map natural language expressions to the KG - in this case to its relations. However, unlike in entity linking, where entities are expressed by entity-specific noun phrases, relations are typically expressed by noun and verb phrase *patterns* that use less specific words. For instance, in our example NLQ, the relations ex:writer and ex:bornIn are explicitly referred to by the phrases "X's writer" and "the birthplace of X", respectively. However, depending on the KG schema, relations may also need to be inferred, like in *"Which Americans have been on the moon?"*, where "Americans" specifies an additional constraint ?person ex:bornIn ex:USA and ex:bornIn is never explicitly mentioned in the question.

3. **Identifying Logical/Numerical Operators**: Sometimes questions contain additional operators on intermediate variables/sets. For example, "*How many writers worked on Westworld*?" implies a COUNT operation on the set of Westworld writers. Other operators can include ARGMAX/MIN, comparative filters (e.g. "older than 40"), set inclusion (e.g. "*Did Michael Crichton write Westworld?"*) etc. Like entities and relations, identifying such operators is primarily a lexicon learning problem. However, compared to entities and relations, there is a fixed small set of operators, which depends on the chosen formal language and not on the KG.

4. **Determining Logical Form Structure**: In order to arrive at a concrete logical form, a series of decisions must be made regarding the structure of the logical form, i.e. how to arrange the operators, relations and entities such that the resulting object executes to the intended answer. For example, if FunQL is used as target formal language, a

decision must be taken to supply ex:Westworld as the argument to the relation function ex:writer, and to pass this subtree as the argument to ex:bornIn, yielding the final query ex:bornIn(ex:writer(ex:Westworld)). Note that such decisions are heavily interdependent with the previous tasks and as mentioned at the beginning of the section, the different types of decisions are not necessarily taken separately. In fact, often structural decisions are merged with or implied by lexical decisions, for example, in FunQL, generating an ARGMAX token at a certain time step implies that the next generated token will be the root of its (ARGMAX's) first argument. On the other hand, the REDUCE operation in transition-based Stack-LSTM semantic parsers Cheng et al. (2017, 2019) is a purely structure-manipulating token that indicates the finalization of a subtree in FunQL.

Question answering often solve a number of these subtasks in a single process. For instance, translation based systems (see Section 4.3) in principle could generate the whole query in a single sequence decoding process, thus solving all subtasks at once. However, in practice, specialized modules can be employed, for example for entity linking, in order to constrain the search space of the main model.

3 Datasets

Research in the field of KGQA has seen a shift from manual feature engineering based solutions (Unger et al., 2015; Berant et al., 2013; Reddy et al., 2014; Höffner et al., 2017) to neural network based, data driven approaches. One of the pivotal requirements for these data-driven approaches is the availability of large datasets comprising of a wide variety of NLQs-label pairs. As a consequence, one can observe a shift in the scale, and more recently, also in the complexity of questions in KGQA datasets. The properties of the most popular KGQA datasets are summerized in Table 1.

One of the first attempts to create a large scale KGQA dataset was by Cai and Yates (2013) who developed the Free917 dataset, consisting out of 917 question / formal query

[8]https://github.com/hobbit-project/QuestionAnsweringBenchmark

Dataset	KG	Size	Formal Queries	Complex Questions
Free917 (Cai and Yates, 2013)	Freebase	917	Yes	Yes
WebQuestions (Berant et al., 2013)	Freebase	5810	No	Yes
WebQuestionsSP (Yih et al., 2016)	Freebase	4737	Yes	Yes
ComplexQuestions (Bao et al., 2016)	Freebase	2100	Yes	Yes
GraphQuestions (Su et al., 2016)	Freebase	5166	Yes	Yes
SimpleQuestions (Bordes et al., 2015)	Freebase	100K	Yes	No
30M Factoid Questions (Serban et al., 2016)	Freebase	30M	Yes	No
QALD [8]	DBpedia	50-500	Yes	Yes
LC-QuAD (Trivedi et al., 2017)	DBpedia	5000	Yes	Yes
LC-QuAD 2 (Dubey et al., 2019)	DBpedia, Wikidata	30 000	Yes	Yes

Table 1: (i) the underlying knowledge graph on which the dataset is based (ii) Complexity of the question (Simple/Complex) (iii) Number of questions (iv) Availability of Formal Query.

pairs covering over 600 Freebase relations. Along the same lines, Berant et al. (2013) developed another dataset called WebQuestions by finding questions using the Google Suggest API and answering them with the help of Amazon Mechanical Turk (AMT) workers using Freebase. Although this dataset is much larger than Free917, it suffers from two disadvantages. First, it does not provide formal queries as targets for NLQs but only question-answer pairs, inhibiting supervised training of KGQA models which rely on a logical form. Second, it is primarily composed of simple questions, and relatively few require complex reasoning. In order to alleviate the first issue Yih et al. (2016) introduced WebQuestionSP, a subset of WebQuestions, having both answers as well as formal queries corresponding to each question. In order to provide more structural variance and expressiveness in questions, Bao et al. (2016) and Su et al. (2016) have released ComplexQuestions and GraphQuestions, respectively, consisting of pairs of questions and their formal queries, where they augment a subset of WebquestionSP with type constraints, implicit and explicit temporal constraints, aggregation operations etc. Trivedi et al. (2017) released LC-QuAD, a dataset of complex questions for the DBpedia KG. They started by generating formal queries for DBpedia and semi-automatically verbalized them using question templates, and then leveraged crowdsourcing to convert these template-based questions to NLQs, performing paraphrasing and grammar correction in the process. Dubey et al. (2019) used a similar mechanism to

create a significantly larger and varied dataset - LC-QuAD 2. QALD[9] (Usbeck et al., 2018) is another important KGQA dataset based on DBpedia. Although it is substantially smaller than LC-QuAD, it has more complex and colloquial questions as they have been created directly by domain experts.

Most datasets discussed so far, due to their relatively small size, do not provide a thorough coverage of the entities and relations that exists in a KG. In order to partially compensate for this and support a larger variety of relations, Bordes et al. (2015) created the SIMPLE QUESTION dataset with more that 100k questions. These questions can be answered by a single triple in the KG, and thus the corresponding formal queries can be thought to simply consist of the subject entity and predicate of the triple. Serban et al. (2016) synthetically expanded the aforementioned dataset to create the 30M FACTOID QUESTIONS dataset consisting of 30 million question-answer pairs.

4 Neural Network based KGQA Systems

As explained in Section 2.3, the problem of KGQA is usually cast into a semantic parsing problem. A semantic parser is an algorithm that given a context (i.e. a KG $\mathcal{K}$ and formal target language with expressions $\mathcal{F}$), maps a given NLQ q to the logical form $f \in \mathcal{F}$, which (1) when executed over $\mathcal{K}$ returns the correct answer a and (2) accurately captures the meaning of q. Neural network-based *semantic parsing algorithms* use prediction *models*, with trainable parameters, that enable the parser to be fitted on a given dataset. The models are trained using a suitable *training procedure*, which depends on the model, the parsing algorithm and the data provided for training the parser.

In this article, we divide the prediction models commonly used in semantic parsing into three major categories, namely: (i) classification, (ii) ranking, and (iii) translation. In the following subsections (Sections 4.1, 4.2, and 4.3, respectively), we will explain each of these models and provide an overview of the previously proposed KGQA systems which employ these different models. We also discuss these models and how are they used and trained in the defined context.

[9]QALD is an ongoing KGQA challenge with multiple tracks - see http://qald.aksw.org/.

In all the training procedures discussed below, the models are optimised with stochastic gradient descent (SGD) or one of its variants. Depending on the type of model, different loss functions can be used. Moreover, depending on the semantic parsing algorithm, different ways of processing and using the provided data in the optimization algorithm are required in order to train the models. These choices define the different training procedures that are discussed in the subsequent sections. Another important aspect is the type of training data which is provided, leading to two different training settings: the *fully supervised* setting, where the dataset consists of N pairs of NLQs and formal queries $\left\{\left(q^{(i)}, f^{(i)}\right)\right\}_{i=1}^{N}$, and the *weakly supervised* setting, where the semantic parser is trained over a dataset of pairs of NLQs and corresponding execution results $\left\{\left(q^{(i)}, a^{(i)}\right)\right\}_{i=1}^{N}$. Whereas the prediction models themselves are generally not affected by this difference, it does lead to different training procedures.

While the fully supervised setting allows to train the models by simply maximizing the likelihood of predicting the correct logical form, the weakly supervised setting presents a more challenging scenario, where we must indirectly infer and "encourage" the (latent) logical forms that execute to the correct answer while avoiding spurious logical forms which might hurt generalization. This gives rise to two fundamental challenges. Firstly, finding *consistent* logical forms (those which execute to the ground truth answer) is challenging due to the size of the search space which usually grows exponentially with respect to the number of tokens in the logical form. Secondly, dealing with spurious candidates, that is, incorrect logical forms (those that do not capture the meaning of the source question) that coincidentally execute to the correct answer, thereby misleading the supervision signal provided to the semantic parser (Cheng and Lapata, 2018).

4.1 Classification based KGQA

In the most general case, the semantic parser should be able to generate a structured output (i.e. the corresponding formal query) of arbitrary size and complexity given the input NLQ. In some cases, however, we can assume a fixed structure for the formal query. This holds in the case of single fact based questions (e.g. SimpleQuestions), where only a single subject entity and a relation need to be predicted. For instance, the question *"Where was Michael*

Crichton born?" is represented in our example KG (Fig. 1) by a single triple pattern based SPARQL query with the triple pattern being $\langle$ex:Michael_Crichton, ex:bornIn, ?x$\rangle$. The logical form corresponding to this type of question thus always consists of one subject entity and a relation, and the answer is given by the missing object entity. For such fixed-structure prediction tasks, we can make use of simple text classification methods to predict the different parts of the target formal query given the NLQ as input (see also Hakimov et al. (2019) for a recent overview of deep learning architectures for simple questions).

4.1.1 Classification Models

To illustrate such classification methods, let us consider the task of relation classification. Given an NLQ q, and a KG $\mathcal{K}$, the relation classification task consists of predicting which of the n_r relations $r_1, \ldots, r_{n_r} \in P_{\mathcal{K}}$ is referred to in q. In the first step, an *encoder network* can be used to map the variable-length input sequence of q onto a fixed-length vector $\mathbf{q} \in \mathbb{R}^d$, which is called *the latent representation*, or the *encoded representation* of q. The encoder network can be comprised of different neural architectures, including recurrent neural networks (RNN) (Hochreiter and Schmidhuber, 1997), convolutional neural networks (CNN) (LeCun and Bengio, 1998), or transformers (Vaswani et al., 2017). The encoded question is subsequently fed through an affine transformation to calculate a *score vector* $s(q) = (s_1(q) \ldots s_{n_r}(q))$, as follows:

$$s(q) = W_o \mathbf{q} + \mathbf{b_o} \ . \tag{2}$$

Here, $W_o \in \mathbb{R}^{n_r \times d}$ and $\mathbf{b_o} \in \mathbb{R}^{n_r}$, in conjunction with the parameters of the encoder network, constitute the trainable parameters of the classification model. In the output layer the classification model typically turns the score vector into a conditional probability distribution $p(r_k|q)$ over the n_r relations based on a softmax function, that is

$$p(r_k|q) = \frac{e^{s_k(q)}}{\sum_{j=0}^{n_r} e^{s_j(q)}} \ , \tag{3}$$

for $k = 1, \ldots n_r$. Classification is then performed by picking the relation with the highest probability given q.

Given a data set of N pairs of NLQs and single fact based formal queries $\mathcal{D} = \left\{\left(q^{(i)}, f^{(i)}\right)\right\}_{i=1}^{N}$, (for SimpleQuestions, $f^{(i)}$ is a entity-relation tuple: $f^{(i)} = (e^{(i)}, r^{(i)})$), the relation classification model is trained by maximising the log-likelihood of the model parameters θ, which is given by

$$\sum_{i=1}^{N} \log p_\theta(r^{(i)}|q^{(i)}) \ , \tag{4}$$

where $r^{(i)}$ is the predicate used in the formal query $f^{(i)}$.

4.1.2 Classification based Parsing Algorithms

For single-fact based prediction tasks, systems relying on standard classification models as described above, can achieve state-of-the-art performance (Mohammed et al., 2018; Petrochuk and Zettlemoyer, 2018). Since the target formal queries consist only of one subject entity and one relation, they can, in principle, be predicted using two separate classifiers, receiving the NLQ as input and producing an output distribution over all entities and all relations in the KG, respectively. This approach can be successfully applied to predict the KG relation mentioned or implied in the question. However, large KGs like Freebase contain a huge amount of entities and training datasets can only cover a small fraction of these (and therefore many classes remain unseen after training). This makes it difficult to apply the approach described above for relation classification to entity prediction.[10] Therefore, several works on SimpleQuestions initially only perform relation classification and rely on a two-step approach for entity linking instead. In such an two-step approach, first an entity span detector[11] is applied to identify the entity mention in the NLQ. Secondly, simple text based retrieval methods can be used to find a limited number of suitable candidate entities. The best fitting entity given q can then be chosen from this candidate set based on simple string similarity and graph-based features. The entity span detector can be implemented based on classifiers as well, either as a pair of classifiers (one for predicting the start position and one

[10] Relation prediction might suffer from the same problem as entity prediction. However, usually there are much fewer relations (thousands) than entities (millions) in a KG, such that a dataset of substantial size can provide several instances of each (important) relation.

[11] The entity span detector does not need to learn representations for entities (like encoder networks used in classifiers), avoiding the need for data covering all entities.

for predicting the end position of the span) or as a collection of independent classifiers[12] (one for every position in the input, akin to a token classification network), where each classifier predicts whether the token at the corresponding position belongs to the entity span or not.

To give a concrete example of a full KGQA semantic parsing algorithm relying on classification models, we will describe the approach proposed by Mohammed et al. (2018), chosen due to its simplicity and competitive performance on SimpleQuestions. This QA system follows the following steps for predicting the entity-relation pair constituting the formal query:

1. Entity span detection: a bidirectional Long-Short Term Memory network (BiLSTM) (Hochreiter and Schmidhuber, 1997) is used for sequence tagging, i.e. to identify the span a of the NLQ q that mentions the entity. In the example "*Where was Michael Crichton born?*", the entity span could be "*Michael Crichton*".

2. Entity candidate generation: Given the entity span, all KG entities are selected whose labels (almost) exactly match the predicted span.

3. Relation classification: A bidirectional gated recurrent unit (BiGRU) (Cho et al., 2014) encoder is used to encode q. The final state of the BiGRU is used to compute probabilities over all relations using a softmax output layer, as described in Equations (2) and (3).

4. Logical form construction: The top-scoring entities and relations from previous steps are taken, and all their possible combinations are ranked based on their component scores and other heuristics. The top-scoring pair is then returned as the predicted logical form of the question.

For training the full system, the relations given in the formal queries of the dataset are used as correct output classes. For the span detector, we first need to extract "pseudo-gold" entity spans since they are not given as part of the dataset. This is done by automatically aligning NLQs with the entity label(s) of the correct entity provided in the training data,

[12] representing a sequence tagger

i.e. the part of the NLQ that best matches the entity label is taken as the correct span to train the sequence tagging BiLSTM.

Mohammed et al. (2018) investigate different RNN and convolutional neural network (CNN) based relation detectors as well as bidirectional long-short-time-memory (LSTM) (Hochreiter and Schmidhuber, 1997) and CRF-based entity mention detectors. They show that a simple BiLSTM-based sequence tagger can be used for entity span prediction leading to results competitive to the state-of-the-art.

Both Petrochuk and Zettlemoyer (2018) and Mohammed et al. (2018) first identify the entity span, similarly to previous works, but disambiguate the entity without using neural networks. Petrochuk and Zettlemoyer (2018) employed a BiLSTM-CRF model for span prediction, which can no longer be considered a simple collection of independent classifiers and instead forms a structured prediction model that captures dependencies between different parts of the output formal query (in this case the binary labels for every position).

4.2 Ranking based KGQA

If we cannot assume that all formal queries follow a fixed structure, as we could in the previous section, the task of mapping an NLQ to its logical form becomes more challenging. Since the set $\mathcal{F}$ of all possible formal queries grows exponentially with the length of the formal query, KGQA systems need to reduce the set of possible outputs. Ranking based semantic parsers therefore employ some kind of search procedure to find a suitable set of candidate formal queries $\mathcal{C}(q) = \{f_1, ..., f_N\}$ for a given NLQ q and the underlying KG, and then use a neural network based ranking model to find the best matching candidate.

4.2.1 Ranking Models

Formally, given an NLQ q and a candidate set of formal queries $\mathcal{C}(q)$, the task of a ranking model is to output a score for each of the candidate formal queries in $\mathcal{C}(q)$ that allows to rank the set of candidate formal queries, where a higher score indicates a better fit for the given NLQ q. In a neural ranking setting, this is accomplished via a two step process. In the first step, a neural encoder model is employed to map q as well as every candidate $f \in \mathcal{C}(q)$ is

mapped to a latent representation space, resulting in the vector $\mathbf{q}$ for the NLQ and vectors $\mathbf{f}$ for every candidate. In the second step, each formal query encoding vector $\mathbf{f}$, paired with the encoded question $\mathbf{q}$ is fed into a differentiable scoring function, that returns a matching score $s\,(q, f)$ indicating how well the formal query f matches the question q. The scoring function $s(\cdot, \cdot)$ can, for example, be a parameterless function such as the dot product between the embedding vectors, or by a parameterized function, such as another neural network receiving the embeddings as input. The highest scoring formal query candidate is then returned as the prediction of the model:

$$f^* = \arg\max_{f \in \mathcal{C}(q)} s_\theta\,(q, f) \quad . \tag{5}$$

When multiple answers are possible for a given question, the following equation may be used to determine a set of outputs by also considering candidates whose scores are close to the best-scoring candidate (Dong et al., 2015):

$$\mathcal{F}_a^* = \{f \mid f \in \mathcal{C}(q) \;\; and \;\; s_\theta(q, f^*) - s_\theta(q, f) < \gamma\} \;\; . \tag{6}$$

We can train ranking models using (i) *full supervision*, where the training data consists of questions and corresponding logical forms, and (ii) *weak supervision*, where gold logical forms are absent, and only pairs of questions and corresponding answers (i.e. execution results from the KG) are available for training. We discuss these two cases below.

Full Supervision Given a dataset $\mathcal{D}^+ = \{\big(q^{(i)}, f^{(i)}\big)\}_{i=1}^N$ of pairs of NLQs and their corresponding *gold* formal queries, a set of *negative examples* $\mathcal{D}^- = \{(q^{(i)}, \hat{f}^{(i)})\}_{i=1}^N$ is generated. That is, for each NLQ $q^{(i)}$ in the training set, a false formal query $\hat{f}^{(i)}$ that does not execute to the correct answer for $q^{(i)}$ is created. Typically, this process, referred to as *negative sampling*, relies on a random or heuristic-based search procedure.

Training the ranking model then corresponds to fitting the parameters of the score function and the neural encoders by minimising either a pointwise or pairwise ranking objective function. A popular ranking objective is the *pairwise hinge loss* that can be computed over pairs of positive and negative examples:

$$\sum_{(q,f) \in \mathcal{D}^+} \sum_{(q,\hat{f}) \in \mathcal{D}^-} \max(0, s_\theta(q, \hat{f}) - s_\theta(q, f) + \gamma) \;\; , \tag{7}$$

Optimizing the model parameters θ by minimizing this function incentives the model to score positive examples higher than negative ones, where $\gamma > 0$ specifies the width of the margin between the scores of positive and negative examples. Alternatively, ranking models can be trained in a pointwise manner by computing the logistic loss over individual positive and negative examples.

Weak Supervision In order to train the ranking objectives under weak supervision, given a dataset of pairs of questions and corresponding execution answers $\mathcal{D}^+ = \{\left(q^{(i)}, a^{(i)}\right)\}_{i=1}^N$, a search procedure is used to find *pseudo gold* logical forms $\mathcal{F}_p^{(i)} = \{f | f\left(\mathcal{K}\right) = a^{(i)}\}$ for each question $q^{(i)}$ that evaluate to the correct answer $a^{(i)}$. Pairs of questions and their corresponding *pseudo gold* logical forms are then used to train the ranking models as described above.

The maximum margin reward (MMR) method introduced by (Peng et al., 2017) proposed a modified formulation of the pairwise hinge-loss based ranking objective in Eq. 7 called the most-violation margin objective. For each training example (q, a), they find the highest scoring logical form f^* from $\mathcal{F}_p$, as the *reference* logical form. They define a reward-based margin function $\delta : \mathcal{F} \times \mathcal{F} \times \mathcal{A} \rightarrow \mathbb{R}$ and find the logical form that violates the margin the most:

$$\hat{f} = \arg\max_{f \in \mathcal{F}} \left(s_\theta(q, f) - s_\theta(q, f^*) + \delta(f^*, f, a)\right) \tag{8}$$

where $\delta(f^*, f, a) = R(f^*, a) - R(f, a)$, and $R(f, a)$ is a scalar-valued reward that can be computed by comparing the labelled answers a and the answers $a' = f(\mathcal{K})$ yielded by executing f on the KG. The reward function chosen by Peng et al. (2017) is the F_1 score. The most-violation margin objective for the training example (q, a) is thus defined as:

$$\max(0, s_\theta(q, \hat{f}) - s_\theta(q, f^*) + \delta(f^*, \hat{f}, a)) \ , \tag{9}$$

where $\hat{f}$ is computed using Eq. 8. Note that the MMR method only updates the score of the reference logical form and the most-violating logical form. MMR essentially generalizes the pairwise hinge loss for non-binary reward functions.

Below, we discuss how existing approaches design the scoring function $s_\theta(q, f)$, followed by a discussion on the parsing algorithms used for finding candidate logical forms in ranking based methods.

Encoding Methods and Score Functions Neural network based models used for encoding questions and logical forms vary in complexity ranging from simple embedding based models to recurrent or convolutional models.

Bordes et al. (2014) proposed one of the first neural network based KGQA methods for answering questions corresponding to formal queries involving multiple entities and relations, and evaluate their method on WebQuestions. They represent questions as bag-of-words vectors of tokens in the question, and logical forms as bag-of-words vectors indicating whether a certain relation or entity is present in the logical form. These sparse bag-of-words representations are then encoded using a neural embedding model that computes the sum of the embeddings for the words, entities and relations present in the bag-of-words vector, yielding fixed-length vector representations for the question and logical form respectively. Their match is scored by computing the dot product between the resulting vectors.

Dong et al. (2015) improve upon the simple embedding-based approach by introducing multi-column CNNs that produce three different vector-based representations of the NLQ. For each NLQ representation, they create three different vector representations for the logical form by encoding (i) the path of relations between the entity mentioned in the question and the answer entity, (ii) the 1-hop subgraph of entities and relations connected to the path, and (iii) the answer type information respectively, using embedding-based models similar to the above method. The sum of dot products between the question representations and their corresponding logical form representations is computed to get the final score.

Zhang et al. (2016) propose an attention-based representation for NLQs and incorporate structural information from the KG into their ranking-based KGQA model by embedding entities and relations using a pretrained TransE model (Bordes et al., 2013). They adopt a multi-task training strategy to alternatingly optimize the TransE and KGQA objectives respectively.

The use of structural information, such as entity type vectors, and pretrained KG embeddings[13] has also been shown to be beneficial for the KGQA task in Dai et al. (2016) and Lukovnikov et al. (2017). Furthermore, Lukovnikov et al. (2017) explore building question representations on both word and character level. Note that these approaches focus on a

[13] A good introduction to such latent feature or KG embedding methods is given by Nickel et al. (2016).

subset of the task, namely answering simple questions with a single entity and relation, and correspondingly train their models on SimpleQuestions. These, and other works on simple questions typically encode the subject entity, and the relations in the logical form separately and rank them against the given NLQ, instead of treating them together as a formal query language expression. An exception to this is Bordes et al. (2015) which ranks entire triples.

Instead of incorporating structural information from the KG, Luo et al. (2018) incorporate local semantic features into the NLQ representation. To do so, they extract the dependency path between the entity mentioned in the NLQ and annotated *wh-token*[14], and encode both the dependency path as well as the NLQ tokens.

Yih et al. (2015) use CNNs to encode graph-structured logical forms called *query graphs* after flattening the graph into a sequence of tokens. Their method is extended by Yu et al. (2017) who use bidirectional LSTMs for encoding the logical forms. They use two kinds of embeddings for encoding relations: relation-specific vector representations, as well as word-level vector representations for the tokens in the relation.

The approaches discussed above are implemented over the Freebase knowledge graph. Maheshwari et al. (2019) propose an attention-based method to compute different representations of the NLQ for each relation in the logical form, and evaluate their approach on LC-QuAD and QALD-7. They also show that transfer learning across KGQA datasets is an effective method of offsetting the general lack of training data, by pre-training their models on LC-QuAD, and fine-tuning on QALD. Additionally, their work demonstrates that the use of pre-trained language models (Devlin et al., 2019; Radford et al., 2019; Yang et al., 2019) for KGQA is a potentially beneficial technique for further increasing the model's performance. Lukovnikov et al. (2019) also explores the use of pre-trained language models for the KGQA task, over SimpleQuestions (which uses Freebase).

4.2.2 Ranking based Parsing Algorithms

As mentioned above, searching for candidate logical forms is a pivotal step in training and inference of ranking-based parsing algorithms. During inference, a search procedure is used to create the formal query candidate set $\mathcal{C}(q)$, which is then ranked using the scoring function.

[14] "what", "where", "when", "who", "which" or "how"

The size of the search space crucially depends on the complexity of the NLQs the systems aims to answer. In the weakly supervised setting, in the absence of gold annotated formal queries, an additional search procedure is required during training to *guess* the latent formal queries that resolve to the correct answer for a given NLQ. A similar search process may be used for generating the negative examples needed for training ranking models.

Typically, the search procedure builds on multiple techniques to restrict the candidate space, such as:

(i) using off-the-shelf entity-linking tools to generate entity candidates, and limiting the search space to formal queries containing only entities and relations found within a certain distance (usually measured in number of relation traversals, also referred to as *hops*) of the candidate entities,

(ii) limiting the number of relations used in the formal query,

(iii) employing *beam search*, a heuristic search algorithm that maintains a *beam* of size K, i.e. a set of at most K incomplete output sequences that have high probability under the current model; at every time step during prediction, every sequence in the beam is expanded with all possible next actions or tokens, and the probabilities of the expanded sequences are evaluated by the scoring function, following which only the K most likely sequences are retained in the beam and considered for further exploration,

(iv) strict pruning of invalid or redundant logical forms that violate grammar constraints of the formal language, or those that do not adhere to the structure of the KG, along with additional rules and heuristics.

Bordes et al. (2014) and Dong et al. (2015) employ (i) and (iii), i.e. they find the entities mentioned in the question, and use a beam search mechanism to iteratively build logical forms one prediction at a time, using their ranking model. In contrast, Xu et al. (2016) propose a multi-stage approach in order to better handle questions with multiple constraints. They use syntactical rules to break a complex question down into multiple simple questions, and use multi-channel CNNs to jointly predict entity and relation candidates in the simple fact-based questions. The predictions of the CNN are validated by textual evidence extracted

from unstructured Wikipedia pages of the respective entities. They train an SVM rank classifier (Joachims, 2006) to choose the best entity-relation pair for each simple question.

Yih et al. (2015) employ a heuristic reward function to generate logical forms using a multi-step procedure, starting by identifying the entities mentioned in the NLQ, then building a *core chain* consisting only of relations that lead to the answer entity, and finally adding constraints like aggregation operators to their logical form. Their logical forms, called *query graphs*, can be viewed as the syntax tree of a formal query in λ-DCS[15] (Liang, 2013). For a given NLQ, partial query graphs generated at each stage of the search procedure are flattened into a sequence of tokens. The NLQ and the formal query tokens are respectively encoded by a CNN model and the dot product of the output representations is used as the final output score.

4.3 Translation based KGQA

In this section, we will focus on methods that learn to *generate* a sequence of tokens that forms the logical form as opposed to learning to choose the correct logical form among a set of pre-generated candidates.

Semantic parsing in this setup is modelled as a translation problem, where we need to translate a NLQ q into a formal query f that can be expected to return the intended answer when executed over the source KG $\mathcal{K}$. A popular approach for mapping sequences in one language to sequences in another language is to use neural sequence-to-sequence (seq2seq) models. Such models were first introduced for machine translation, i.e., for mapping a sentence in one natural language to the corresponding sentences in another (Bahdanau et al., 2014; Luong et al., 2015). With some changes, this neural architecture has been extensively adapted for semantic parsing (Dong and Lapata, 2016; Jia and Liang, 2016; Guo et al., 2018a; Zhong et al., 2017; Xu et al., 2017; Sun et al., 2018) and more specifically for the KGQA task (He and Golub, 2016; Liang et al., 2017; Yin et al., 2018; Cheng and Lapata, 2018; Guo et al., 2018b).

[15]Concretely, the core chain can be described as a series of conjunctions in λ-DCS, where the peripheral paths of the tree are combined using intersection operators.

4.3.1 Translation Models

A typical neural sequence-to-sequence model consists of an encoder, a decoder and an attention mechanism. The encoder encodes the input sequence to create context-dependent representations for every token in the input sequence. The decoder generates the output sequence, one token at a time, conditioning the generation process on previously generated tokens as well as the input sequence. The attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) models the alignment between input and output sequences which has proven to be a very useful inductive bias for translation models. In neural sequence-to-sequence models, generally RNNs are used to encode the input and to predict the output tokens. Other encoder/decoder network choices are also possible, such as CNNs and transformers (Vaswani et al., 2017). Since logical forms are generally tree-structured, and a basic sequence decoder does not explicitly exploit the tree dependencies, several works have focused on developing more structured decoders (see Section 4.3.1).

Formally, given an input sequence $q_{0...T}$[16] of T tokens from the input vocabulary $\mathcal{V}^I$ and an output sequence $f_{0...T^*}$ of T^* tokens from the output vocabulary $\mathcal{V}^O$, the translation model with parameters θ, models the probability $p_\theta(f_{0...T^*}|q_{0...T})$ of generating the whole output sequence given the input sequence. The probability of the whole output sequence can be decomposed into the product of probabilities over the generated tokens

$$p_\theta(f_{0...T^*}|q_{0...T}) = \prod_{j=0}^{T^*} p_\theta(f_j|f_{<j}, q_{0...T}) \ , \tag{10}$$

where $f_{<j}$ is the sequence of tokens generated so far, i.e. $f_{<j} = (f_0, \ldots, f_{j-1})$.

During prediction, the output sequences are usually generated by taking the most likely token at every time step, which is also referred to as *greedy search*, that is

$$f_j = \arg\max_{f \in \mathcal{V}^O} p_\theta(f|f_{<j}, q_{0..T}) \ , \tag{11}$$

or by using *beam search* (see Section. 4.2.2).

In the following, we discuss common ways to train translation based KGQA models in the fully and weakly supervised settings.

[16] We use the notation $q_{0...T}$ to denote $(q_0, \ldots, q_T)$, the sequence of symbols q_i, for $i \in [0, T]$

Full Supervision In the *fully supervised* setting, the parameters θ of the model $p_\theta(f_j|f_{<j}, q_{0..T})$ are typically trained by maximizing their likelihood given the sequence pairs in the training set $\mathcal{D} = \left\{\left(q^{(i)}, f^{(i)}\right)\right\}_{i=1}^{N}$, that is, by maximizing the following objective:

$$\sum_{(q,f)\in\mathcal{D}} p_\theta(f_{0...T^*}|q_{0...T}) \ . \tag{12}$$

Weak Supervision In the *weakly supervised* setting, the training data $\mathcal{D} = \{\left(q^{(i)}, a^{(i)}\right)\}_{i=1}^{N}$ consists only of pairs of NLQs and corresponding execution results. Different methods have been proposed for training translation models for semantic parsers using only weak supervision. Even though the proposed methods vary in the objective function and training procedure, they all operate on the general principle of maximizing the probability of producing the correct execution results.

The following training methods are commonly used for weakly supervising translation-based semantic parsing models:

1. **Maximum Marginal Likelihood (MML):** The maximum marginal likelihood method follows a similar idea as the maximum likelihood (ML) method used in the fully supervised setting, but instead of directly maximizing the probability of producing the correct logical form, it maximizes the probability of producing the correct execution results. In order to do so, MML marginalizes over all possible logical forms, maximizing the following objective function:

$$\sum_{(q,a)\in\mathcal{D}} \log p(a|q) = \sum_{(q,a)\in\mathcal{D}} \log \sum_{f\in\mathcal{F}} p(a|f)p_\theta(f|q) \ , \tag{13}$$

   where a is the correct answer to the NLQ q and the sum is computed over all f's in the space of all possible logical forms $\mathcal{F}$. Since semantic parsing environments are usually deterministic (i.e. the same query always executes to the same answer when the KG is kept fixed), the $p(a|f)$ term is either 1 (if f's execution produced the correct results a, i.e. $f\mathcal{K}) = a$) or 0, which leads to the following simplified notation

$$\sum_{(q,a)\in\mathcal{D}} \log \sum_{f\in\mathcal{F}^*} p_\theta(f|q) \ , \tag{14}$$

where $\mathcal{F}^*$ is the set of consistent logical forms that execute to the correct answer $a^{(i)}$. The set $\mathcal{F}^*$ is usually approximated using online beam search. However, an approximation of $\mathcal{F}^*$ can also be computed beforehand (Pasupat and Liang, 2016) and kept fixed throughout training.

2. **Reinforcement Learning using Expected Rewards (ER):** The problem of weakly supervised semantic parsing can also be approached as a reinforcement learning (RL) problem. In fact, we can view the translation model as a parameterized policy that given a state, which in our case consists of the KG $\mathcal{K}$, input q and decoding history $f_{<t}$, must decide what action to take in order to maximize a reward. In our case, the reward function $R(f, a)$ can be a binary reward at the end of an episode (end of decoding phase), that is 1 if the produced trajectory (logical form) f executes to the correct answer a and 0 otherwise. The RL setup for semantic parsing is characterized by (1) a deterministic environment (the next state produced by executing the action on the current state is always the same) and (2) sparse rewards (the reward is given only at the end of the episode and, given the huge number of possible logical forms, is likely to be 0 most of the time). In addition, weakly supervised training of semantic parsing is characterized by *underspecified* rewards (Agarwal et al., 2019) which could lead to learning *spurious* logical forms. However, the reward function based on execution results only cannot take this into account.

   Policy gradient methods are trained by optimizing the *expected reward*:

$$\sum_{(q,a)\in\mathcal{D}} \sum_{f\in\mathcal{F}} R(f, a) p_\theta(f|q) \ , \tag{15}$$

   where the sum is over all possible logical forms, which, in practice, is estimated using an approximation strategy such as *Monte Carlo integration*, i.e. based on trajectories sampled from the policy.

3. **Iterative Maximum Likelihood (IML):** The Iterative Maximum Likelihood (IML) objective (Liang et al., 2017) uniformly maximizes the probability of decoding *all*

consistent logical forms across all examples:

$$\sum_{(q,a)\in\mathcal{D}} \sum_{f\in\mathcal{F}^*} \log\ p_\theta(f|q)\ , \tag{16}$$

When training policy gradient methods using ER, (Liang et al., 2017) demonstrate that better exploration can be achieved by employing IML to populate an initial buffer of diverse trajectories (i.e. logical forms) which are then used to pretrain the RL model.

4. **MAPO:** In order to reduce the variance of the ER estimator, Liang et al. (2018) proposed a novel method that integrates a memory buffer. They reformulated the ER objective as a sum of action sequences in a memory buffer $\mathcal{C}(q)$ and outside the buffer:

$$\mathcal{O}_{MAPO}() = \sum_{(q,a)\in\mathcal{D}} \sum_{f\in\mathcal{C}(q)} R(f,a)p_\theta(f|q) + \sum_{f\notin\mathcal{C}(q)} R(f,a)p_\theta(f|q)\ . \tag{17}$$

The memory buffers $\mathcal{C}(q)$ for each example i are populated using *systematic exploration*, which prevents revisiting sequences that have already been explored. When the two terms are explored further (see Liang et al. (2018)), we can see that the weight assigned to trajectories from the memory buffer is low in the beginning of training, when the policy still assigns low probabilities to the buffered trajectories. In order to speed up training, Liang et al. (2018) propose *memory weight clipping*, which amounts to assigning the buffered trajectories an importance weight of at least a certain value. Experimental results of Liang et al. (2018) show significant improvement of the proposed MAPO procedure against common baselines and show that both systematic exploration and memory weight clipping are essential to achieve high performance.

Agarwal et al. (2019) argue that apart from an optional entropy term, MAPO does not encourage exploration, which can be problematic. They propose a MAPO variant called MAPOX where the initial buffer is additionally populated by sequences found using IML.

A disadvantage of the RL and MML approaches, noted by Guu et al. (2017) and Agarwal et al. (2019), is that the exploration of trajectories is guided by the current model policy, which means that logical forms that have high probability under the current model are more likely to be explored. Consequently, logical forms with low probability may be overlooked

by exploration. In the presence of many spurious logical forms and only a few correct logical forms, it becomes more likely that spurious ones are explored first. Once the policy settles on these high-reward (but spurious) logical forms, the exploration gets increasingly biased towards them during training, leading to poor generalization. One common solution to this is to perform ϵ-greedy exploration, which Guu et al. (2017) extend from RL to MML. Guu et al. (2017) also propose a *meritocratic update rule* that updates parameters such that probability is spread more evenly across consistent logical forms. For a more detailed discussion of the MML and ER objectives, we refer the reader to the work of Guu et al. (2017), Norouzi et al. (2016), Roux (2017) and Misra et al. (2018).

Agarwal et al. (2019) proposes a meta-learning approach for learning an auxiliary reward function that is used to decrease the effect of spurious logical forms in the MAPOX objective. The auxiliary reward function is trained such that the update to the policy under the reward augmented by the auxiliary reward function results in better generalization over a held-out batch of data.

Cheng and Lapata (2018) propose a neural parser-ranker based approach for weakly supervised semantic parsing, where a sequence-to-sequence *parsing* model (trained with beam search and IML) is coupled with a basic ranking model (trained using MML) as well as an *inverse parsing* model, which is a generative model that reconstructs the original question from the logical form generated by the *parsing model*. The reconstruction loss is used to further refine the parsing model in order to tackle the problem of spurious logical forms. Note that here the logical form is treated as a latent variable.

Another class of neural approaches for KGQA with weak supervision perform multi-step knowledge base reasoning in a fully differentiable end-to-end manner. TensorLog is a recently-developed differentiable logic that performs approximate first order logic inference through a sequence of differentiable numerical operations on matrices. NeuralLP (Yang et al., 2017), inspired by TensorLog, learns to map an input question to its answer by performing multi-step knowledge base reasoning by means of differentiable graph traversal operations.

The neural programmer (Neelakantan et al., 2015) is a fully differentiable encoder-decoder based architecture augmented with a set of manually defined discrete operators (e.g. argmax, count), which is applied to the task of table-based complex question answering by Neelakan-

tan et al. (2017). The discrete operators allow the model to induce *latent logical forms* by composing arithmetic and logical operations in a differentiable manner. The model is trained using a weak supervision signal which is the result of the execution of the correct program.

In contrast, the Neural Symbolic Machines framework (Liang et al., 2017) combines (i) a differentiable neural *programmer*, which is a seq2seq model augmented by a key-variable memory that can translate a natural language utterance to a program, and (ii) a symbolic *computer* (specifically, a Lisp interpreter) that implements a domain-specific language with built-in functions and code-assistance that is used to prune syntactically or semantically invalid canddiate logical forms and execute them to retrieve a result that is used to compute the supervision signal. REINFORCE, augmented with *iterative maximum likelihood training*, is used to optimize for rewarding programs. Liang et al. (2018) demonstrate the effectiveness of a memory buffer of promising programs, coupled with techniques to stabilize and scale up training as well as reduce variance and bias.

Model Variations Several extensions have been proposed to the standard sequence-to-sequence model that try to improve the translation model for semantic parsing in general, as well as some more task- or dataset-specific extensions.

Semantic parsers with **structured decoders** use the same sequence encoders to encode the NLQ but induce additional structure on top of the normal attention-based sequence decoder that exploits the hierarchical tree structure of the query.

Dong and Lapata (2016) propose a tree decoding model that decodes a query tree in a top-down breadth-first order. For example, instead of decoding the lambda-calculus logical form `(argmin $0 (state:t $0) (size:i $0))` (corresponding to the question "*Which state is the smallest?*") as a sequence, the decoder first decodes the topmost level of the query tree `(argmin $0 <n><n></s>)`. Here, `<n>` and `</s >` are artificial topological tokens introduced to indicate the tree structure: `<n>` is a non-terminal token indicating that a subtree is expected at its position and `</s>` is the end-of-sequence token indicating the end of a sequence of siblings. After decoding this top-level sequence, the two children subtrees, `(state:t $0)` and `(size:i $0)`, are decoded by conditioning them on the the two non-terminal states. From experimental results on GEO880, ATIS, JOBQUERIES and IFTTT,

it appears that the inductive bias introduced by the tree decoder improves generalization.

Alvarez-Melis and Jaakkola (2016) propose an improved tree decoder, where the parent-to-child and sibling-to-sibling information flows are modeled with two separate RNNs. With this model, each node has a parent state and a previous-sibling state, both of which are used to predict the node symbol and topological decisions. Instead of modeling topological decisions (i.e. whether a node has children or further siblings) through artificial topological tokens like Dong and Lapata (2016), they use auxiliary classifiers at every time step that predict whether a node has children and whether the node is the last of its siblings. The elimination of artificial topological tokens reduces the length of the generated sequences, which should lead to fewer errors. The decoding proceeds in a top-down breadth-first fashion, similarly to Dong and Lapata (2016). Experimental results on the IFTTT semantic parsing dataset show improvement obtained by the introduced changes.

Cheng et al. (2017; 2018; 2019) develop a transition-based neural semantic parser that adapts the Stack-LSTM proposed by Dyer et al. (2015). The Stack-LSTM decodes the logical forms in a depth-first order, decoding a subtree completely before moving on to its siblings. The Stack-LSTM of Cheng and Lapata (2018) uses an adapted LSTM update when a subtree has been completed: it backtracks to the parent state of the completed subtree, and computes a summary encoding of the completed subtree and uses it as input in the next LSTM update. Cheng et al. (2019) also shows how to perform bottom-up transition-based semantic parsing using the Stack-LSTM. Cheng et al. (2017) obtains state-of-the-art results on GraphQuestions and Spades and results competitive with previous works on WebQuestions and GeoQueries. Cheng and Lapata (2018) further improves performance of their Stack-LSTM model on the weakly supervised GraphQuestions, Spades and WebQuestions datasets by using a generative ranker.

Copying Mechanisms (Vinyals et al., 2015; See et al., 2017; Gu et al., 2016; Jia and Liang, 2016) are an augmentation of the seq2seq neural architecture which enables a direct copy of tokens or sub-sequences from the input sequences to the output. Though it (Vinyals et al., 2015; See et al., 2017; Gu et al., 2016; Jia and Liang, 2016) is not generally required for semantic parsing, it can be useful for certain tasks or datasets. In WikiSQL, for example, a copying mechanism is required to copy SQL condition values into the query (Shi et al.,

2018; Xu et al., 2017; Zhong et al., 2017). It has also been used for semantic parsing in general (Jia and Liang, 2016; Damonte et al., 2019).

Another important aspect is **symbol representation**. When answering questions over large scale knowledge graphs, we are confronted with a large number of entities and relations not present in the training data. An important challenge, thus, is to find a way to learn representations for entities and relations that generalize well to unseen ones. Representing both entities and predicates as a sequence of words, characters or sub-word units (BPE/WordPiece) in their *surface forms*, as opposed to arbitrary symbols unique to each entity or predicate, offers an extent of generalizability to these unseen or rare symbols. For instance, if the representations are learned on sub-word levels, upon encountering sub-words from unseen relations, the parameters of the representation building network (encoder) can leverage sub-words shared between seen and unseen relations and thus, unseen relations will no longer have uninformed random vector representations.

Several works on question answering and semantic parsing (and other NLP tasks) have used such computed representations. For example, Several works on WikiSQL also encode column names on word level to yield vectors representing columns (Shi et al., 2018; Sun et al., 2018; Yu et al., 2018). Some works on KGQA (Yu et al., 2017; Lukovnikov et al., 2017; Maheshwari et al., 2019; He and Golub, 2016) also decompose KG relations and/or entities to word and/or sub-word level. In addition to sub-token level encoding, additional information about the tokens can be encoded and added to its representation. For example, Yu et al. (2018) also encodes table names and column data types together with column name words for their model for the Spider dataset.

4.3.2 Translation based Parsing Algorithms

In the case of purely translation-based semantic parsing, the *parsing algorithm* centers around sequence decoding, which is usually done using greedy search or beam search as explained above.

Constrained Decoding A simple sequence-to-sequence model as described above does not exploit the formal nature of the target language. In fact, the output sequences to be

generated follow strict grammatical rules that ensure the validity of the decoded expression, i.e. that they can be executed over the given KG. Thus, many translation based parsing algorithms exploit these grammatical rules by using grammar-based constraints during decoding in order to generate only valid logical forms. Using such constraints during training also help to focus learning and model capacity on the space of valid logical forms. Depending on the specific application and dataset, reasonable assumptions concerning the logical form can be made. These choices are reflected in the logical form language definition. As an example, in the case of SimpleQuestions, the assumption is that the logical from only consists of single entity and a single relation. Therefore, if we were to solve SimpleQuestions using a translation model, we could constrain the decoder to choose only among all entities in the first time step and only among all relations in the second (instead of considering the full output vocabulary of all tokens possible in the formal query language) automatically terminate decoding thereafter.

For more general cases, we would like to express a wider range of formal query structures, but nevertheless apply some restrictions on the output tokens at a certain time step, depending on the sequence decoded so far. For example, in the FunQL language definition used by Cheng et al. (2017, 2019), the $\textsc{argmax}(x, r)$ function is restricted to have a relation symbol r as second argument. Constrained decoding can be trivially implemented by just considering the set of allowed tokens as possible outputs at a certain time step. Computing the allowed tokens for a certain time step however, can be more challenging, depending on the chosen logical form language.

Enhanced Decoding Procedures The two-stage Coarse2Fine decoder of Dong and Lapata (2018) can be seen as a middle-ground between a sequence decoder and a tree decoder. The decoder consists of two decoding stages: (1) decoding a query template and (2) filling in the specific details. Compared to other tree decoders, the Coarse2Fine decoder also proceeds in a top-down breadth-first manner, but is restricted to have only two levels. For cases when there is a limited number of query templates, Dong and Lapata (2018) also investigate the use of a template classifier (instead of decoding the template) in the first decoding stage and evaluate on WikiSQL. An additional improvement to the two-step decoding scheme

is obtained by encoding the generated template using a bidirectional RNN, before using the output states to decode the details. This allows to condition the generation of specific arguments of the template on the whole structure of the template.

The work of Cheng and Lapata (2018) trains and uses a translation model for semantic parsing. Using beam search, several logical forms are decoded from the translation model and additional ranking models (see Section 4.3.1) are used to re-rank the logical forms in the beam.

5 Emerging Trends

Question answering over knowledge graphs has been an important area of research in the past decade. them being the use of query graph candidates (Yih et al., 2015; Yu et al., 2017; Maheshwari et al., 2019), the use of neural symbolic machines (Liang et al., 2017), the shift to answering multi-entity questions (Luo et al., 2018), the application of transfer learning (Maheshwari et al., 2019), and the proposal of differentiable query execution based weakly supervised models (Yang et al., 2017). Petrochuk and Zettlemoyer (2018) suggest that the performance on SimpleQuestions is approaching an upper bound. Further, as discussed in Section 3, there is a general shift of focus towards more complex logical forms, as is evident by recent datasets like (Trivedi et al., 2017; Bao et al., 2016; Dubey et al., 2019). These advances lay a path for further improvements in the field, and the base for the emerging trends and challenges we outline in the following.

Query Complexity Comparative evaluations over WebQuestions, a dataset over Freebase, demonstrate the rise of performance of KGQA approaches over the years on the task (Bao et al., 2016). Recently Petrochuk and Zettlemoyer (2018) demonstrate "*that ambiguity in the data bounds* [the] *performance at 83.4% [over* SimpleQuestions*]*" thereby suggesting that the progress over the task is further ahead than ordinarily perceived. For instance, the previously best performing baseline proposed by Yu et al. (2017) of 77% accuracy over SimpleQuestions, can be perceived as 92.3% (of 83.4) instead. Given that in WebQuestions where several KGQA systems have shown high performance, about 85% of the quesitons are simple questions as well (Bao et al., 2016), similar claims may be hold

in this context, pending due investigations.

Knowledge graphs commonly used in KGQA community, as well as the formal query languages used to facilitate KGQA can support more nuanced information retrieval involving longer core chains, multiple triples, joins, unions, and filters etc. These nuanced retrieval mechanisms are increasingly being regarded in the community as the next set of challenges. Recently released datasets such as ComplexQuestions (Bao et al., 2016), GraphQuestions (Su et al., 2016), LC-QuAD (Trivedi et al., 2017), and LC-QuAD 2 (Dubey et al., 2019) explicitly focus on creating complex questions, with aggregates, constraints, and longer relation paths, over which the current systems do not perform very well, and there is a possibility of significant improvements.

Robustness: In the past few years, deep learning based system have achieved state of the art performance over several tasks, however, an increasing number of findings points out the brittleness of these systems. For instance, Jia and Liang (2017) demonstrate a drop of 35%-75% in F1 scores of sixteen models for the reading comprehension task trained over SQuAD (Rajpurkar et al., 2016), by adversarially adding another sentence to the input paragraph (from which the system has to select the relevant span, given the question). Following, a new version of the aforementioned dataset was released comprising of *unanswerable* questions (Rajpurkar et al., 2018), leading to more robust reading comprehension approaches (Hu et al., 2018; Kundu and Ng, 2018). To the best of our knowledge, there hasn't been any work quantifying or improving the robustness of KGQA models. Such advances, would play an important role for the applicability of KGQA systems in a production setting. In our opinion, a good starting point for robustness inquiries would be to utilize recent general purpose adversarial input frameworks. Ribeiro et al. (2018) propose a simple, generalisable way to generate semantically equivalent adversarial sentences.

Interoperability between KGs: In discussing the numerous KGQA approaches in the previous section, we find that only a handful of techniques include datasets from both DBpedia and Freebase in their experiments, despite both of them being general purpose KGs. This is because the inherent data model underlying the two KGs differs a lot, which makes extracting *(question, DBpedia answer)* pairs corrsponding to a set of *(question, Freebase answer)* pairs (or vice versa) nontrivial, even after discounting the engineering efforts

spent in migrating the query execution, and candidate generation sub-systems. This is best illustrated in Tanon et al. (2016) where different hurdles and their solutions of migrating the knowledge from Freebase to Wikidata is discussed. Following a similar approach, Diefenbach et al. (2017) migrate the SimpleQuestions dataset to Wikidata, yielding 21,957 answerable questions over the KG. Azmy et al. (2018) migrate it to DBpedia (October, 2016 release), and provide 43,086 answerable questions.

Correspondingly, interoperability of KGQA systems is another challenge, which only recently has drawn some attention in the community (Abbas et al., 2016). The two fold challenge of (i) learning to identify multiple KG's artifacts mentioned in a given NLQ, and (ii) learning multiple parse structures corresponding to the multiple KG's data models, while very difficult (Ringler and Paulheim, 2017), is partially helped by the latest (upcoming) DBpedia release[17], whose data model is compatible with that of Wikidata's. For a in-depth discussion on knowledge modeling strategies, and comparison of major largescale open KGs, we refer interested readers to (Ismayilov et al., 2018; Ringler and Paulheim, 2017; Färber et al., 2018).

Multilinguality: Multilinguality, i.e. the ability to understand and answer questions in multiple languages is pivotal for a widespread acceptance and use of KGQA systems. With varying coverage, large parts of common knowledge graphs including DBpedia, Wikidata have multilingual surface forms corresponding to the resources, bridging a major challenge in enabling multilinguality in KGQA systems.

The QALD challenge, currently in its 9th iteration maintains multilingual question answering as one of its tasks. The dataset accompanying QALD-9[18] multilingual QA over DBpedia task contains over 250 questions in upto eight languages including English, Spanish, German, Italian, French, Dutch, Romanian, Hindi and Farsi. (Diefenbach et al., 2018) propose a non-neural *generate and rank* approach with minimal language dependent components which can be replaced to support new languages. (Radoev et al., 2018) propose using a set of multilingual lexico-syntactic patterns to understand the intent of both French and English questions. However, we still have to see a surge of multilinguality in data driven KGQA

[17] http://downloads.dbpedia.org/repo/lts/wikidata/

[18] https://project-hobbit.eu/challenges/qald-9-challenge/#tasks

approaches. Since these approaches rely on supervised data to learn mappings between KG artifacts, and question tokens; the lack of large scale, multilingual, KGQA datasets inhibits these approaches.

6 Concluding Remarks

Answering questions over knowledge graphs has emerged as an multi-disciplinary field of research, inviting insights and solutions from the semantic web, machine learning, and the natural language understanding community. In this article, we provide an overview of neural network based approaches for the task.

We broadly group existing approaches in three categories namely, (i) classification based approaches, where neural models are used to predict one of a fixed set of classes, given a question, (ii) ranking based approaches, where neural networks are used to compare different candidate logical forms with the question, to select the best-ranked one, and (iii) translation based where the network learns to translate natural language questions (NLQs) into their corresponding (executable) logical forms. Along with an overview of existing approaches, we also discuss some techniques used to weakly supervise the training of these models, which cope with the challenges risen due to a lack of logical forms in the training data. We summarize existing datasets and tasks commonly used to benchmark these approaches, and note that the progress in the field has led to performance saturation over existing datasets such as SimpleQuestions, leading to an emergence of newer, more difficult challenges, as well as more powerful mechanisms to address these challenges.

Towards the end of the article, we discuss some of the emerging trends, and existing gaps in the KGQA research field, concluding that investigations and innovations in interoperability, multilinguality, and robustness of these approaches are required for impactful application of these systems.

7 Acknowledgements

We acknowledge support by the European Union H2020 grant Cleopatra (GA no. 812997).

References

Abbas, F., Malik, M. K., Rashid, M. U., and Zafar, R. (2016). Wikiqa a question answering system on wikipedia using freebase, dbpedia and infobox. *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*, pages 185–193.

Ackermann, W. and Hilbert, D. (1928). Grundzüge der theoretischen logik. *Berlin, Springcr*, 1037(23):4.

Agarwal, R., Liang, C., Schuurmans, D., and Norouzi, M. (2019). Learning to generalize from sparse and underspecified rewards. *CoRR*, abs/1902.07198.

Alvarez-Melis, D. and Jaakkola, T. S. (2016). Tree-structured decoding with doubly-recurrent neural networks.

Azmy, M., Shi, P., Lin, J., and Ilyas, I. F. (2018). Farewell freebase: Migrating the simplequestions dataset to dbpedia. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 2093–2103.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bao, J., Duan, N., Yan, Z., Zhou, M., and Zhao, T. (2016). Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514.

Barendregt, H. P. (1984). Introduction to lambda calculus.

Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors (2018). *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*.

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Bordes, A., Chopra, S., and Weston, J. (2014). Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620.

Bordes, A., Usunier, N., Chopra, S., and Weston, J. (2015). Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Cai, Q. and Yates, A. (2013). Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 423–433.

Cheng, J. and Lapata, M. (2018). Weakly-supervised neural semantic parsing with a generative ranker. In *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018, Brussels, Belgium, October 31 - November 1, 2018*, pages 356–367.

Cheng, J., Reddy, S., Saraswat, V., and Lapata, M. (2017). Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 44–55.

Cheng, J., Reddy, S., Saraswat, V., and Lapata, M. (2019). Learning an executable neural semantic parser. *Computational Linguistics*, 45(1):59–94.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Moschitti, A., Pang, B., and Daelemans, W., editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

Church, A. (1936). An unsolvable problem of elementary number theory. *American journal of mathematics*, 58(2):345–363.

Dai, Z., Li, L., and Xu, W. (2016). Cfo: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 800–810.

Damonte, M., Goel, R., and Chung, T. (2019). Practical semantic parsing for spoken language understanding. *arXiv preprint arXiv:1903.04521*.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Diefenbach, D., Both, A., Singh, K. D., and Maret, P. (2018). Towards a question answering system over the semantic web. *CoRR*, abs/1803.00832.

Diefenbach, D., Tanon, T. P., Singh, K. D., and Maret, P. (2017). Question answering benchmarks for wikidata. In *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017*.

Dong, L. and Lapata, M. (2016). Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 33–43.

Dong, L. and Lapata, M. (2018). Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 731–742.

Dong, L., Wei, F., Zhou, M., and Xu, K. (2015). Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 260–269.

Dubey, M., Banerjee, D., Abdelkawi, A., and Lehmann, J. (2019). Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *International Semantic Web Conference*. Springer.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Färber, M., Bartscherer, F., Menne, C., and Rettinger, A. (2018). Linked data quality of dbpedia, freebase, opencyc, wikidata, and YAGO. *Semantic Web*, 9(1):77–129.

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *J. Artif. Intell. Res.*, 57:345–420.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Guo, D., Sun, Y., Tang, D., Duan, N., Yin, J., Chi, H., Cao, J., Chen, P., and Zhou, M. (2018a). Question generation from sql queries improves neural semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1597–1607.

Guo, D., Tang, D., Duan, N., Zhou, M., and Yin, J. (2018b). Dialog-to-action: Conversational question answering over a large-scale knowledge base. In Bengio et al. (2018), pages 2946–2955.

Guu, K., Pasupat, P., Liu, E. Z., and Liang, P. (2017). From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1051–1062.

Hakimov, S., Jebbara, S., and Cimiano, P. (2019). Evaluating architectural choices for deep learning approaches for question answering over knowledge bases. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 110–113. IEEE.

He, X. and Golub, D. (2016). Character-level question answering with attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1598–1607.

Hirschman, L. and Gaizauskas, R. (2001). Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Höffner, K., Walter, S., Marx, E., Usbeck, R., Lehmann, J., and Ngonga Ngomo, A.-C. (2017). Survey on challenges of question answering in the semantic web. *Semantic Web*, 8(6):895–920.

Hu, M., Wei, F., Peng, Y., Huang, Z., Yang, N., and Zhou, M. (2018). Read + verify: Machine reading comprehension with unanswerable questions. *CoRR*, abs/1808.05759.

Ismayilov, A., Kontokostas, D., Auer, S., Lehmann, J., and Hellmann, S. (2018). Wikidata through the eyes of dbpedia. *Semantic Web*, 9(4):493–503.

Jia, R. and Liang, P. (2016). Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 12–22.

Jia, R. and Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2021–2031.

Joachims, T. (2006). Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.

Kamath, A. and Das, R. (2019). A survey on semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1051–1062.

Kate, R. J. and Mooney, R. J. (2006). Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 913–920. Association for Computational Linguistics.

Kundu, S. and Ng, H. T. (2018). A nil-aware answer extraction framework for question answering. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J., editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4243–4252. Association for Computational Linguistics.

LeCun, Y. and Bengio, Y. (1998). The handbook of brain theory and neural networks. chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. MIT Press, Cambridge, MA, USA.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2015). DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195. Outstanding Paper Award (Best 2014 SWJ Paper).

Liang, C., Berant, J., Le, Q., Forbus, K. D., and Lao, N. (2017). Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 23–33.

Liang, C., Norouzi, M., Berant, J., Le, Q. V., and Lao, N. (2018). Memory augmented policy optimization for program synthesis and semantic parsing. In Bengio et al. (2018), pages 10015–10027.

Liang, P. (2013). Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*.

Lukovnikov, D., Fischer, A., and Lehmann, J. (2019). Pretrained transformers for simple question answering over knowledge graphs. In *International Semantic Web Conference*. Springer.

Lukovnikov, D., Fischer, A., Lehmann, J., and Auer, S. (2017). Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pages 1211–1220. International World Wide Web Conferences Steering Committee.

Luo, K., Lin, F., Luo, X., and Zhu, K. (2018). Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2185–2194.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Maheshwari, G., Trivedi, P., Lukovnikov, D., Chakraborty, N., Fischer, A., and Lehmann, J. (2019). Learning to rank query graphs for complex question answering over knowledge graphs. In *International Semantic Web Conference*. Springer.

Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. (2011). Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM.

Misra, D., Chang, M.-W., He, X., and Yih, W.-t. (2018). Policy shaping and generalized update equations for semantic parsing from denotations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2442–2452.

Mohammed, S., Shi, P., and Lin, J. (2018). Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 291–296.

Neelakantan, A., Le, Q. V., Abadi, M., McCallum, A., and Amodei, D. (2017). Learning a natural language interface with neural programmer. In *International Conference on Learning Representations*.

Neelakantan, A., Le, Q. V., and Sutskever, I. (2015). Neural programmer: Inducing latent programs with gradient descent. *CoRR*, abs/1511.04834.

Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. (2016). A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Norouzi, M., Bengio, S., Chen, Z., Jaitly, N., Schuster, M., Wu, Y., and Schuurmans, D. (2016). Reward augmented maximum likelihood for neural structured prediction. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1723–1731.

Pasupat, P. and Liang, P. (2016). Inferring logical forms from denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 23–32.

Peng, H., Chang, M.-W., and Yih, W.-t. (2017). Maximum margin reward networks for learning from explicit and implicit supervision. In *Proceedings of the 2017 Conference*

on Empirical Methods in Natural Language Processing, pages 2368–2378, Copenhagen, Denmark. Association for Computational Linguistics.

Petrochuk, M. and Zettlemoyer, L. (2018). Simplequestions nearly solved: A new upper-bound and baseline approach. *arXiv preprint arXiv:1804.08798*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Radoev, N., Zouaq, A., Tremblay, M., and Gagnon, M. (2018). A language adaptive method for question answering on french and english. In Buscaldi, D., Gangemi, A., and Recupero, D. R., editors, *Semantic Web Challenges - 5th SemWebEval Challenge at ESWC 2018, Heraklion, Greece, June 3-7, 2018, Revised Selected Papers*, volume 927 of *Communications in Computer and Information Science*, pages 98–113. Springer.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Reddy, S., Lapata, M., and Steedman, M. (2014). Large-scale semantic parsing without question-answer pairs. *Transactions of the Association of Computational Linguistics*, 2(1):377–392.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 856–865.

Ringler, D. and Paulheim, H. (2017). One knowledge graph to rule them all? analyzing the differences between dbpedia, yago, wikidata & co. In *KI 2017: Advances in Artificial Intelligence - 40th Annual German Conference on AI, Dortmund, Germany, September 25-29, 2017, Proceedings*, pages 366–372.

Roux, N. L. (2017). Tighter bounds lead to improved classifiers. In *International Conference on Learning Representations*.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Serban, I. V., García-Durán, A., Gulcehre, C., Ahn, S., Chandar, S., Courville, A., and Bengio, Y. (2016). Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598. Association for Computational Linguistics.

Shi, T., Tatwawadi, K., Chakrabarti, K., Mao, Y., Polozov, O., and Chen, W. (2018). Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *CoRR*, abs/1809.05054.

Su, Y., Sun, H., Sadler, B., Srivatsa, M., Gur, I., Yan, Z., and Yan, X. (2016). On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572.

Sun, Y., Tang, D., Duan, N., Ji, J., Cao, G., Feng, X., Qin, B., Liu, T., and Zhou, M. (2018). Semantic parsing with syntax-and table-aware sql generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 361–372.

Tanon, T. P., Vrandecic, D., Schaffert, S., Steiner, T., and Pintscher, L. (2016). From freebase to wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 1419–1428.

Trivedi, P., Maheshwari, G., Dubey, M., and Lehmann, J. (2017). Lc-quad: A corpus for complex question answering over knowledge graphs. In *International Semantic Web Conference*, pages 210–218. Springer.

Unger, C., Forascu, C., López, V., Ngomo, A. N., Cabrio, E., Cimiano, P., and Walter, S. (2015). Question answering over linked data (QALD-5). In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015.*

Usbeck, R., Gusmita, R. H., Ngomo, A. N., and Saleem, M. (2018). 9th challenge on question answering over linked data (QALD-9) (invited paper). In *Joint proceedings of the 4th Workshop on Semantic Deep Learning (SemDeep-4) and NLIWoD4: Natural Language Interfaces for the Web of Data (NLIWOD-4) 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.*, pages 58–64.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Xu, K., Reddy, S., Feng, Y., Huang, S., and Zhao, D. (2016). Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2326–2336.

Xu, X., Liu, C., and Song, D. (2017). Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Yang, F., Yang, Z., and Cohen, W. W. (2017). Differentiable learning of logical rules for knowledge base reasoning. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M.,

Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2316–2325.

Yang, Y. and Chang, M.-W. (2016). S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. *arXiv preprint arXiv:1609.08075*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Yih, W.-t., Chang, M.-W., He, X., and Gao, J. (2015). Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1321–1331.

Yih, W.-t., Richardson, M., Meek, C., Chang, M.-W., and Suh, J. (2016). The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 201–206.

Yin, P., Zhou, C., He, J., and Neubig, G. (2018). Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 754–765.

Yu, M., Yin, W., Hasan, K. S., dos Santos, C., Xiang, B., and Zhou, B. (2017). Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 571–581.

Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z., and Radev, D. (2018). Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1653–1663.

Zhang, Y., Liu, K., He, S., Ji, G., Liu, Z., Wu, H., and Zhao, J. (2016). Question answering over knowledge base with neural attention combining global knowledge information. *arXiv preprint arXiv:1606.00979*.

Zhong, V., Xiong, C., and Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.