Leveraging Abstract Meaning Representation for Knowledge Base Question Answering

Pavan Kapanipathi[*], Ibrahim Abdelaziz[*], Srinivas Ravishankar[*], Salim Roukos, Alexander Gray, Ramon Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, Mo Yu
IBM Research

Abstract

Knowledge base question answering (KBQA) is an important task in Natural Language Processing. Existing approaches face significant challenges including complex question understanding, necessity for reasoning, and lack of large end-to-end training datasets. In this work, we propose Neuro-Symbolic Question Answering (NSQA), a modular KBQA system, that leverages (1) Abstract Meaning Representation (AMR) parses for task-independent question understanding; (2) a simple yet effective graph transformation approach to convert AMR parses into candidate logical queries that are aligned to the KB; (3) a pipeline-based approach which integrates multiple, reusable modules that are trained specifically for their individual tasks (semantic parser, entity and relationship linkers, and neuro-symbolic reasoner) and do not require end-to-end training data. NSQA achieves state-of-the-art performance on two prominent KBQA datasets based on DBpedia (QALD-9 and LC-QuAD 1.0). Furthermore, our analysis emphasizes that AMR is a powerful tool for KBQA systems.

1 Introduction

Knowledge base question answering (KBQA) is a sub-field within Question Answering with desirable characteristics for real-world applications. KBQA requires a system to answer a natural language question based on facts available in a Knowledge Base (KB) (Zou et al., 2014; Vakulenko et al., 2019; Diefenbach et al., 2020; Abdelaziz et al., 2021). Facts are retrieved from a KB through structured queries (in a query language such as SPARQL), which often contain multiple triples that represent the steps or antecedents required for obtaining the answer. This enables a transparent and self-explanatory form of QA, meaning that intermediate symbolic representations capture some of the steps from natural language question to answer.

With the rise of neural networks in NLP, various KBQA models approach the task in an end-to-end manner. Many of these approaches formulate text-to-query-language as sequence-to-sequence problem, and thus require sufficient examples of paired natural language and target representation pairs. However, labeling large amounts of data for KBQA is challenging, either due to the requirement of expert knowledge (Usbeck et al., 2017), or artifacts introduced during automated creation (Trivedi et al., 2017). Real-world scenarios require solving complex multi-hop questions i.e. secondary unknowns within a main question and questions employing unusual expressions. Pipeline approaches can delegate language understanding to pre-trained semantic parsers, which mitigates the data problem, but are considered to suffer from error propagation. However, the performance of semantic parsers for well-established semantic representations has greatly improved in recent years. Abstract Meaning Representation (AMR) (Banarescu et al., 2013; Dorr et al., 1998) parsers recently reached above 84% F-measure (Bevilacqua et al., 2021), an improvement of over 10 points in the last three years.

In this paper we propose Neuro-Symbolic Question Answering (NSQA), a modular knowledge base question answering system with the following objectives: (a) delegating the complexity of understanding natural language questions to AMR parsers; (b) reducing the need for end-to-end (text-to-SPARQL) training data with a pipeline architecture where each module is trained for its specific sub-task; (c) facilitating the use of an independent reasoner via an intermediate logic form.

[*]Equal contribution, correspondence to Pavan Kapanipathi (kapanipa@us.ibm.com), Ibrahim Abdelaziz (ibrahim.abdelaziz1@ibm.com), Srinivas Ravishankar (srini@ibm.com)

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3884–3894
August 1–6, 2021.

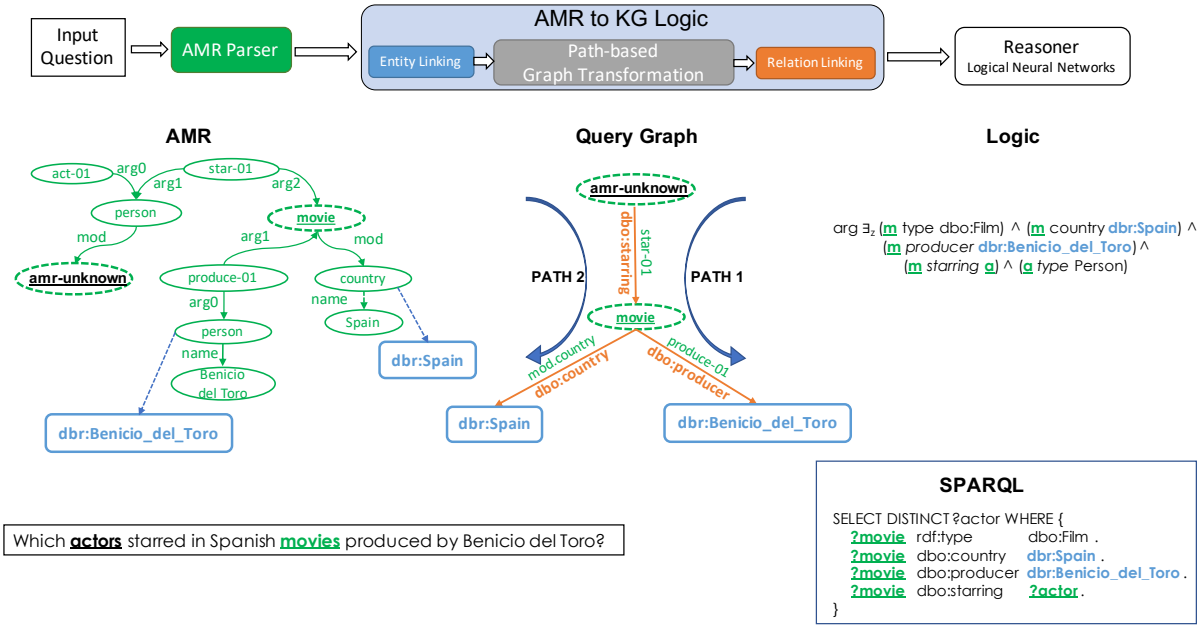

Figure 1: Real NSQA prediction for the sentence *Which actors starred in Spanish movies produced by Benicio del Toro?*. In underlined, we show the representation for the two unknown variables across all stages including: AMR-aligned tokens in sentence (*Which*, *movies*), AMR graph (*unknown*, *movie*), paths representation (same as AMR), logical representation (*actor*, *movie*) and SPARQL interpretation (*?actor*, *?movie*). Displayed stage outputs: AMR (green), Entity Linking (blue), Relation Linking (orange)

The contributions of this work are as follows:

- The first system to use Abstract Meaning Representation for KBQA achieving state of the art performance on two prominent datasets on DBpedia (QALD-9 and LC-QuAD 1.0).
- A novel, simple yet effective path-based approach that transforms AMR parses into intermediate logical queries that are aligned to the KB. This intermediate logic form facilitates the use of neuro-symbolic reasoners such as Logical Neural Networks (Riegel et al., 2020), paving the way for complex reasoning over knowledge bases.
- A pipeline-based modular approach that integrates multiple, reusable modules that are trained specifically for their individual tasks (e.g. semantic parsing, entity linking, and relationship linking) and hence do not require end-to-end training data.

2 Approach Overview

Figure 1 depicts the pipeline of our NSQA system. Given a question in natural language, NSQA: (i) parses questions into an Abstract Meaning Representation (AMR) graph; (ii) transforms the AMR graph to a set of candidate KB-aligned logical queries, via a novel but simple graph transformation approach; (iii) uses a Logical Neural Network (LNN) (Riegel et al., 2020) to reason over KB facts and produce answers to KB-aligned logical queries. We describe each of these modules in the following sections.

2.1 AMR Parsing

NSQA utilizes AMR parsing to reduce the complexity and noise of natural language questions. An AMR parse is a rooted, directed, acyclic graph. AMR nodes represent concepts, which may include normalized surface symbols, Propbank frames (Kingsbury and Palmer, 2002) as well as other AMR-specific constructs to handle named entities, quantities, dates and other phenomena. Edges in an AMR graph represent the relations between concepts such as standard OntoNotes roles but also AMR specific relations such as polarity or mode.

As shown in Figure 1, AMR provides a representation that is fairly close to the KB representation. A special *amr-unknown* node, indicates the missing concept that represents the answer to the given question. In the example of Figure 1, *amr-unknown* is a *person*, who is the subject of *act-01*. Furthermore, AMR helps identify intermediate variables

that behave as secondary unknowns. In this case, a *movie* produced by *Benicio del Toro* in *Spain*.

NSQA utilizes a stack-Transformer transition-based model (Naseem et al., 2019; Astudillo et al., 2020) for AMR parsing. An advantage of transition-based systems is that they provide explicit question text to AMR node alignments. This allows encoding closely integrated text and AMR input to multiple modules (Entity Linking and Relation Linking) that can benefit from this joint input.

2.2 AMR to KG Logic

The core contribution of this work is our next step where the AMR of the question is transformed to a query graph aligned with the underlying knowledge graph. We formalize the two graphs as follows:

AMR graph $\mathcal{G}$ is a rooted edge-labeled directed acyclic graph $\langle \mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G} \rangle$. The edge set $\mathcal{E}_\mathcal{G}$ consists of non-core roles, quantifiers, and modifiers. The vertex set $\mathcal{V}_\mathcal{G} \in$ *amr-unknown* $\cup\, \mathcal{A}_\mathcal{P} \cup \mathcal{A}_\mathcal{C}$ where where $\mathcal{A}_\mathcal{P}$ are set of propbank predicates and $\mathcal{A}_\mathcal{C}$ are rest of the nodes. [1]. Propbank predicates are *n-ary* with multiple edges based on their definitions. *amr-unknown* is a special concept node in the AMR graph indicating *wh-questions*.

Further, we enrich the AMR Graph $\mathcal{G}$ with explicit links to entities in the KG. For example, the question in Figure 1 contains two entities *Spain* and *Benicio Del Toro* that need to be identified and linked to DBpedia entries, *dbr:Spain* and *dbr:Benicio_del_toro*. Linking these entities is absolutely necessary for any KBQA system (Zou et al., 2014; Vakulenko et al., 2019). To do so, we trained a BERT-based neural mention detection model and used BLINK (Devlin et al., 2018) for disambiguation. The entities are linked to AMR nodes based on the AMR node-text alignment information. The linking is a bijective mapping from $\mathcal{V}_e \rightarrow E$ where $\mathcal{V}_e$ is the set of AMR entity nodes, and E is the set of entities in the underlying KG.

Query graph $\mathcal{Q}$ is a directed edge-labeled graph $\langle \mathcal{V}_\mathcal{Q}, \mathcal{E}_\mathcal{Q} \rangle$, which has a similar structure to the underlying KG. $\mathcal{V}_\mathcal{Q} \in \mathcal{V}_\mathcal{E} \cup \mathcal{V}$ where $\mathcal{V}_\mathcal{E}$ is a set of entities in the KG and ($\mathcal{V}$) is a set of unbound variables. $\mathcal{E}_\mathcal{Q}$ is a set of binary relations among $\mathcal{V}_\mathcal{Q}$ from the KG. The Query Graph $\mathcal{Q}$ is essentially the WHERE clause[2] in the SPARQL query.

[1] https://www.isi.edu/~ulf/amr/help/amr-guidelines.pdf

[2] The Query Graph does not include the type constraints in the SPARQL WHERE Clause.

Our goal is to transform the AMR graph $\mathcal{G}$ into its corresponding query graph $\mathcal{Q}$. However such transformation faces the following challenges:

N-ary argument mismatch: Query graph $\mathcal{Q}$ represents information using binary relations, whereas AMR graph contain Propbank framesets that are n-ary. For example, the node *produce-01*[3] from $\mathcal{A}_\mathcal{P}$ in $\mathcal{G}$ has four possible arguments, whereas its corresponding KG relation in $\mathcal{Q}$ (*dbo:producer*) is a binary relation.

Structural and Granular mismatch: The vertex set of the query graph $\mathcal{Q}$ represent entities (or unbound variables). On the other hand, AMR Graph $\mathcal{G}$ contains nodes that are concepts or PropBank predicates which can correspond to both entities and relationships. For example in Figure 1, *produce-01, star-01,* and *Spain* are nodes in the AMR graph. So the AMR graph $\mathcal{G}$ has to be transformed such that nodes primarily correspond to entities and edges (edge labels) correspond to relationships. Furthermore, it is possible for multiple predicates and concepts from $\mathcal{G}$ to jointly represent a single binary relation in $\mathcal{Q}$ because the underlying KG uses a completely different vocabulary. An example of such granular mismatch is shown in Figure 2.

2.2.1 Path-based Graph Transformation

We address the challenges mentioned above by using a path-based approach for the construction of Query Graphs. In KBQA, query graphs (i.e. SPARQL queries) constrain the unknown variable based on paths to the grounded entities. In Figure 1, the constraints in the SPARQL query are based on paths from *?actor* to *dbr:Benicio_del_toro* and *dbr:Spain* as shown below.

- `?actor` → `dbo:starring` → `?movie` → `dbo:country` → `dbr:Spain`
- `?actor` → `dbo:starring` → `?movie` → `dbo:producer` → `dbr:Benicio del Toro`

Based on this intuition of finding paths from the unknown variable to the grounded entities, we have developed a path-based approach depicted in Algorithm 1 that shows the steps for transforming the AMR Graph $\mathcal{G}$ into Query Graph $\mathcal{Q}$. As *amr-unknown* is the unknown variable in the AMR Graph, we retrieve all shortest paths (line 11 in Algorithm 1) between the *amr-unknown* node and the nodes $\mathcal{V}_E$ of the AMR Graph $\mathcal{G}$ that have mappings

[3] http://verbs.colorado.edu/propbank/framesets-english-aliases/produce.html

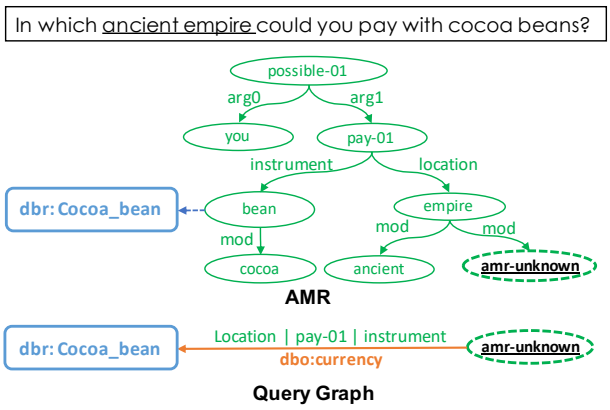

Figure 2: An example of granularity and structural mismatch between AMR and query graphs of the question *In which ancient empire could you pay with cocoa beans?*. The binary relation *dbo:currency* corresponds to the combination of two edges (*location*, *instrument*) and one node (*pay-01*) in the AMR graph.

to the KG entity set. Figure 1 shows an example of both the AMR and query graph for the question "Which actors starred in Spanish movies produced by Benicio del Toro?" Selecting the shortest paths reduces the n-ary predicates of AMR graph to only the relevant binary edges. For instance, the edge *(act-01, arg0, person)* in the AMR graph in Figure 1 will be ignored because it is not in the path between *amr-unknown* and any of the entities *dbr:Spain* and *dbr:Benicio_del_Toro*.

Structural and granularity mismatch between the AMR and query graph occurs when multiple nodes and edges in the AMR graph represent a single relationship in the query graph. This is shown in Figure 2. The path consists of one AMR node and 2 edges between *amr-unknown* and *cocoa bean*: (*amr-unknown*, location, *pay-01*, instrument, *cocoa-bean*)[4]. In such cases, we collapse all nodes that represent predicates (like *pay-01*, *star-01*, etc.) into an edge, and combine it with surrounding edge labels, giving (*location* | *pay-01* | *instrument*). This is done by line 18 of Algorithm 1 where the eventual query graph $\mathcal{Q}$ will have one edge with merged predicated from AMR graph $\mathcal{G}$ between the non-predicates ($\mathcal{A}_C$).

Returning to the example in Figure 1, Algorithm 1 (line 25) outputs the query graph $\mathcal{Q}$ with the following two paths, which bear structural similarity to the knowledge graph:

Algorithm 1: AMR to triples

1 **Input** : Question text t, AMR graph $\mathcal{G} : \langle \mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G} \rangle$ having a set of nodes $\mathcal{V}_e \in \mathcal{V}_\mathcal{G}$, each linked to a named entity in the KG
2 **Returns** : Query graph $\mathcal{Q} : \langle \mathcal{V}_\mathcal{Q}, \mathcal{E}_\mathcal{Q} \rangle$
Initialize $\mathcal{Q} : \langle \mathcal{V}_\mathcal{Q}, \mathcal{E}_\mathcal{Q} \rangle, \mathcal{V}_\mathcal{Q} := \{\}, \mathcal{E}_\mathcal{Q} := \{\}$
3 **if** t *is imperative* **then**
4 let source node (imperative predicate) be r
5 set q as *amr-unknown* where $edge(r, q, \text{'}arg1\text{'}) \in \mathcal{G}_\mathcal{E}$
6 delete r and its edges from $\mathcal{G}$
7 $a :=$ *amr-unknown* node
8 **if** $\exists\ b : edge(a, b, \text{'}mod\text{'}) \in \mathcal{E}_\mathcal{G}$ **then**
9 $a := b$
10 **for** e *in* $\mathcal{V}_e$ **do**
11 $amrPath :=$ **getShortestPath**$(\mathcal{G}, a, e)$
12 let $amrPath$ be $[a, n_1, n_2, ..., n_k]$, where $n_k = e$
13 $collapsedPath := [a]$
14 $n' := a$
15 $relBuilder :=$ ' '
16 **for** $i : 1 \rightarrow k$ **do**
17 **if** $n_i \in \mathcal{A}_\mathcal{P}$ **then**
18 $relBuilder :=$ $relBuilder +$ **getRel**$_G(n', n_i)$
19 **else if** $n_i \in \mathcal{A}_C$ **then**
20 append n_i to $collapsedPath$
21 add node n' to $\mathcal{V}_\mathcal{Q}$
22 add $edge(n', n_i, relBuilder)$ to $\mathcal{E}_\mathcal{Q}$
23 $n' := n_i$
24 $relBuilder :=$ ' '
 end
end
25 $\mathcal{Q} := doRelLinking(\mathcal{Q})$
26 **return** $\mathcal{Q}$

- `amr-unknown → star-01 → movie → country → Spain`
- `amr-unknown → star-01 → movie → produce-01 → Benicio del Toro`

Note that in the above paths, edge labels reflect the predicates from the AMR graph (*star-01*, *produce-01*, and *mod*). Our next step is to resolve these edge labels to its corresponding relationships from the underlying KG. To do so, we perform relation linking as described below.

Relationship Linking. NSQA uses SemREL (Naseem et al., 2021), a state-of-the-art relation linking system that takes in the question text and AMR predicate as input and returns a ranked list of KG relationships for each triple. The cartesian product of this represents a ranked list of candidate query graphs, and we choose the highest-ranked *valid* query graph (a KG subgraph with unbound variables). As shown in Figure 1, the output of this module produces query graph $\mathcal{Q}$ with *star-01* and *produce-01* mapped to DBpedia rela-

[4]Nodes are indicated by boldface, and the rest are edges. For the purposes of path generation, the nodes *amr-unknown* and *empire* are considered equivalent because the *mod* edge is a descriptor in AMR (line 8 in Algorithm 1)

tions *dbo:starring* and *dbo:producer*. This will be the WHERE clause of the final SPARQL query.

2.2.2 Logic Generation

Our query graph can be directly translated to the WHERE clause of the SPARQL. We use existential first order logic (FOL) as an intermediate representation, where the non-logical symbols consist of the binary relations and entities in the KB as well as some additional functions to represent SPARQL query constructs (e.g. COUNT). We use existential FOL instead of directly translating to SPARQL because: (a) it enables the use of any FOL reasoner which we demonstrate in our next Section 2.3; (b) it is compatible with reasoning techniques beyond the scope of typical KBQA, such as temporal and spatial reasoning; (c) it can also be used as a step towards query embedding approaches that can handle incompleteness of knowledge graphs (Ren and Leskovec, 2020; Cohen et al., 2020; Sun et al., 2020). The Query Graph from Section 2 can be written as a conjunction in existential first order logic as shown in Figure 1.

The current logic form supports SPARQL constructs such as SELECT, COUNT, ASK, and SORT which are reflected in the types of questions that our system is able to answer in Table 4. The heuristics to determine these constructs from AMR are as follows:

Query Type: This rule determines if the query will use the ASK or SELECT construct. Boolean questions will have AMR parses that either have no *amr-unknown* variable or have an *amr-unknown* variable connected to a *:polarity* edge (indicating a true/false question). In such cases, the rule returns ASK, otherwise it returns SPARQL.

Target Variable: This rule determines what unbound variable follows a SPARQL statement. As mentioned in Section 2, the *amr-unknown* node represents the missing concept in a question, so it is used as the target variable for the query. The one exception is for questions that have an AMR predicate that is marked as imperative, e.g. in Figure 3 (middle) a question beginning with "Count the awards ..." will have *count-01* marked as imperative. In these cases, the algorithm uses the *arg1* of the imperative predicate as the target variable (see Algorithm 1, line 3).

Sorting: This rule detects the need for sorting by the presence of superlatives and quantities in the query graph prior to relation linking. Superlatives are parsed into AMR with *most* and *least* nodes and quantities are indicated by the PropBank frame *have-degree-91*, whose arguments determine: (1) which variable in $\mathcal{V}$ represents the quantity of interest, and (2) the direction of the sort (ascending or descending).

Counting: This rule determines if the COUNT aggregation function is needed by looking for PropBank frame *count-01* or AMR edge *:quant* connected to *amr-unknown*, indicating that the question seeks a numeric answer. However, questions such as "How many people live in London?" can have *:quant* associated to *amr-unknown* even though the correct query will use *dbo:population* to directly retrieve the numeric answer without the COUNT aggregation function. We therefore exclude the COUNT aggregation function if the KB relation corresponding to *:quant* or *count-01* has a numeric type as its range.

2.3 Reasoner

With the motivation of utilizing modular, generic systems, NSQA uses a First Order Logic, neuro-symbolic reasoner called Logical Neural Networks (LNN) (Riegel et al., 2020). This module currently supports two types of reasoning: type-based, and geographic. Type-based reasoning is used to eliminate queries based on inconsistencies with the type hierarchy in the KB. On the other hand, a question like "Was Natalie Portman born in United States?" requires geographic reasoning because the entities related to *dbo:birthPlace* are generally cities, but the question requires a comparison of countries. This is addressed by manually adding logical axioms to perform the required transitive reasoning for property *dbo:birthPlace*. We wish to emphasize that the intermediate logic and reasoning module allow for NSQA to be extended for such complex reasoning in future work.

3 Experimental Evaluation

The goal of the work is to show the value of AMR as a generic semantic parser on a modular KBQA system. In order to evaluate this, we first perform an end-to-end evaluation of NSQA (Section 3.2). Next, we discuss some qualitative and quantitative results on the value of AMR for different aspects of our KBQA system (Section 3.3). Finally, in support of our modular architecture, we evaluate the individual modules that are used in comparison to other state of the art approaches (Section 3.4).

	Dataset	P	R	F
WDAqua	QALD-9	26.09	26.7	24.99
gAnswer	QALD-9	29.34	**32.68**	29.81
NSQA	QALD-9	**31.89**	32.05	**31.26**
WDAqua	LC-QuAD 1.0	22.00	38.00	28.00
QAMP	LC-QuAD 1.0	25.00	**50.00**	33.00
NSQA	LC-QuAD 1.0	**44.76**	45.82	**44.45**

Table 1: NSQA performance on QALD-9 and LC-QuAD 1.0

3.1 Datasets and Metrics

To evaluate NSQA, we used two standard KBQA datasets on DBpedia.

QALD - 9 (Usbeck et al., 2017) dataset has 408 training and 150 test questions in natural language, from DBpedia version 2016-10. Each question has an associated SPARQL query and gold answer set. Table 4 shows examples of all the question types in the QALD dev set.

LC-QuAD 1.0 (Trivedi et al., 2017) is a dataset with 5,000 questions based on templates and more than 80% of its questions contains two or more relations. Our modules are evaluated against a random sample of 200 questions from the training set. LC-QuAD 1.0 predominantly focuses on the multi-relational questions, aggregation (e.g. COUNT) and simple questions from Table 4.

Dev Set. We also created a randomly chosen development set of 98 QALD-9 and 200 LC-QuAD 1.0 questions for evaluating individual modules.

Metrics. We report performance based on standard precision, recall and F-score metrics for the KBQA system and other modules. For the AMR parser we use the standard Smatch metric (Cai and Knight, 2013).

3.2 End-to-end Evaluation

Baselines: We evaluate NSQA against four systems: GAnswer (Zou et al., 2014), QAmp (Vakulenko et al., 2019), WDAqua-core1 (Diefenbach et al., 2020), and a recent approach by (Liang et al., 2021). GAnswer is a graph data-driven approach and is the state-of-the-art on the QALD dataset. QAmp is another graph-driven approach based on message passing and is the state-of-the-art on LC-QuAD 1.0 dataset. WDAqua-core1 is knowledge base agnostic approach that, to the best of our knowledge, is the only technique that has been evaluated on both QALD-9 and LC-QuAD 1.0 on different versions of DBpedia. Lastly, Liang et al. (Liang et al., 2021) is a recent approach that uses an ensemble of entity and relation linking modules and train a Tree-LSTM model for query ranking.

	AMR3.0	QALD-9	LC-QuAD 1.0
stack-Transformer	80.00	87.91	84.03

Table 2: AMR parsing performance (Smatch) on the AMR3.0 test and QALD-9, LC-QuAD 1.0 dev sets.

Results: Table 1 shows the performance of NSQA compared to state-of-the-art approaches on QALD and LC-QuAD 1.0 datasets. On QALD-9 and LC-QuAD 1.0, NSQA achieves state-of-the-art performance. It outperforms WDAqua and gAnswer on QALD-9. Furthermore, NSQA's performance on LC-QuAD 1.0 significantly outperforms QAmp by 11.45 percentage points on F1.

Due to difference in evaluation setup in Liang et al. (2021), we reevaluated their system on the same setup and metrics as the above systems. Given the test set and the evaluation, (Liang et al., 2021)'s F1 score reduces to 29.2%[5]. We exclude this work from our comparison due to lack of standard evaluation.

3.3 Performance Analysis of AMR

AMR Parsing. We manually created AMRs for the train and dev sets of QALD and LC-QuAD 1.0 questions. The performance of our stack-transformer parser on both of these datasets is shown in Table 2. The parser is trained on the combination of human annotated treebanks and a synthetic AMR corpus. Human annotated treebanks include AMR3.0 and 877 questions sentences (250 QALD train + 627 LC-QuAD 1.0 train sentences) annotated in-house. The synthetic AMR corpus includes 27k sentences obtained by parsing LC-QuAD 1.0 and LC-QuAD 2.0 (Dubey et al., 2019) training sentences, along the lines of (Lee et al., 2020).

AMR-based Query Structure NSQA leverages many of the AMR features to decide on the correct query structure. As shown in Section 2.2.2, NSQA relies on the existence of certain PropBank predicates in the AMR parse such as *have-degree-91*, *count-01*, *amr-unknown* to decide on which SPARQL constructs to add. In addition, the AMR parse determines the structure of the WHERE clause. In Table 3, we show the accuracy of each one of

[5] Liang et al. (2021) report an F1 score of 68% on a different subset of LC-QuAD 1.0. They also consider only questions that returns an answer which is a different setup from the rest of the systems.

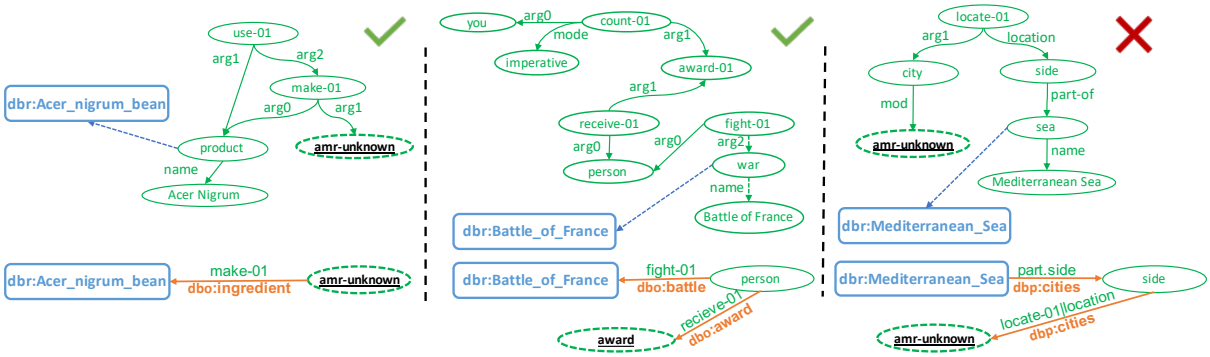

Figure 3: AMR and query graphs for the questions *"Acer nigrum is used in making what?"*, *"Count the awards received by the ones who fought the battle of france?"* and *"What cities are located on the sides of mediterranean sea?"* from LC-QuAD 1.0 dev set

Query Feature	Correct	Total	Correct (%)
SELECT	164	186	88.2
ASK	14	9	64.3
COUNT	25	31	80.6
1-Hop	50	63	79.4
2-Hop	96	137	70.1

Table 3: Query constructs prediction (LC-QuAD 1.0 dev)

these rules on LC-QuAD 1.0 dev dataset. Overall, NSQA identified 64% of ASK (boolean) questions correctly and achieved more than 80% accuracy for COUNT and SELECT questions. Using AMR and the path-based approach, NSQA was able to correctly predict the total number of constraints with comparable accuracies of 79% and 70% for single and two-hops, respectively. NSQA finds the correct query structure for complex questions almost as often as for simple questions, completely independent of the KG.

Figure 3 shows two examples illustrating how AMR lends itself to an intuitive transformation to the correct query graph, as well as a third example where we fail. Here the AMR semantic parse can not be matched to the underlying KG, since 'side' is an extra intermediate variable that leads to an additional constraint in the query graph.

Supported Question Types. Table 4 shows the reasoning and question types supported by NSQA . Our transformation algorithm applied to AMR parses supports simple, multi-relational, count-based, and superlative question types. LNN performs geographic reasoning as well as type-based reasoning to rank candidate logic forms. Addressing comparative and temporal reasoning is a part of our future work.

3.4 Individual Module Evaluation

Entity and Relation Linking. NSQA's EL module (NMD+BLINK) consists of a BERT-based neural mention detection (NMD) network, trained on LC-QuAD 1.0 training dataset comprising of 3,651 questions with manually annotated mentions, paired with an off-the-shelf entity disambiguation model – BLINK (Wu et al., 2019b). We compare the performance of NMD+BLINK approach with Falcon (Sakor et al., 2019) in Table 5. NMD+BLINK performs 24% better on F1 than Falcon (state-of-the-art) on LC-QuAD 1.0 dev set and 3% better on QALD-9 dev set. Similarly, we evaluate Relation Linking on both QALD and LC-QuAD 1.0 dev sets. In particular, we used SemREL (Naseem et al., 2021); state-of-the-art relation linking approach which performs significantly better compared to both Falcon (Sakor et al., 2019) and SLING (Mihindukulasooriya et al., 2020) on various datasets. On LC-QuAD 1.0 dev, SemREL acheives F1 = 0.55 compared to 0.43 by SLING and 0.42 by Falcon. On QALD-9, SemREL achieves 0.54 compared to 0.64 and 0.46 F1 for SLING and Falcon, respectively.

Reasoner. We investigate the effect of using LNN as a reasoner equipped with axioms for type-based and geographic reasoning. We evaluated NSQA's performance under two conditions: (a) with an LNN reasoner with intermediate logic form and (b) with a deterministic translation of query graphs to SPARQL. On LC-QuAD 1.0 dev set, NSQA achieves an F1 score of 40.5 using LNN compared to 37.6 with the deterministic translation to SPARQL. Based on these initial promising results,

Question Type/Reasoning	Example	Supported
Simple	Who is the mayor of Paris	✓
Multi-relational	Give me all actors starring in movies directed by William Shatner.	✓
Count-based	How many theories did Albert Einstein come up with?	✓
Superlative	What is the highest mountain in Italy?	✓
Comparative	Does Breaking Bad have more episodes than Game of Thrones?	
Geographic	Was Natalie Portman born in the United States?	✓
Temporal	When will start [*sic*] the final match of the football world cup 2018?	

Table 4: Question types supported by NSQA , with examples from QALD

	Dataset	P	R	F1
Falcon	QALD-9	0.81	0.83	0.82
NMD+BLINK	QALD-9	0.82	0.90	**0.85**
Falcon	LC-QuAD 1.0	0.56	0.69	0.62
NMD+BLINK	LC-QuAD 1.0	0.87	0.86	**0.86**

Table 5: Performance of Entity Linking modules compared to SOTA Falcon on our dev sets

we intend to explore more uses of such reasoners for KBQA in the future.

4 Related Work

Early work in KBQA focused mainly on designing parsing algorithms and (synchronous) grammars to semantically parse input questions into KB queries (Zettlemoyer and Collins, 2007; Berant et al., 2013), with a few exceptions from the information extraction perspective that directly rely on relation detection (Yao and Van Durme, 2014; Bast and Haussmann, 2015). All the above approaches train statistical machine learning models based on human-crafted features and the performance is usually limited.

Deep Learning Models. The renaissance of neural models significantly improved the accuracy of KBQA systems (Yu et al., 2017; Wu et al., 2019a). Recently, the trend favors translating the question to its corresponding subgraph in the KG in an end-to-end learnable fashion, to reduce the human efforts and feature engineering. This includes two most commonly adopted directions: (1) embedding-based approaches to make the pipeline end-to-end differentiable (Bordes et al., 2015; Xu et al., 2019); (2) hard-decision approaches that generate a sequence of actions that forms the subgraph (Xu et al., 2018; Bhutani et al., 2019).

On domains with complex questions, like QALD and LC-QuAD, end-to-end approaches with hard-decisions have also been developed. Some have primarily focused on generating SPARQL sketches (Maheshwari et al., 2019; Chen et al., 2020) where they evaluate these sketches (2-hop) by providing gold entities and ignoring the evaluation of selecting target variables or other aggregation functions like sorting and counting. (Zheng and Zhang, 2019) generates the question subgraph via filling the entity and relationship slots of 12 predefined question template. Their performance on these datasets show significant improvement due to the availability of these manually created templates. Having the advantage of predefined templates does not qualify for a common ground to be compared to generic and non-template based approaches such as NSQA, WDAqua, and QAmp.

Graph Driven Approaches. Due to the lack of enough training data for KBQA, several systems adopt a training-free approach. WDAqua (Diefenbach et al., 2017) uses a pure rule-based method to convert a question to its SPARQL query. gAnswer (Zou et al., 2014) uses a graph matching algorithm based on the dependency parse of question and the knowledge graph. QAmp (Vakulenko et al., 2019) is a graph-driven approach that uses message passing over the KG subgraph containing all identified entities/relations where confidence scores get propagated to the nodes corresponding to the correct answers. Finally, (Mazzeo and Zaniolo, 2016) achieved superior performance on QALD-5/6 with a hand-crafted automaton based on human analysis of question templates. A common theme of these approaches, is that the process of learning the subgraph of the question is heavily KG specific, while our approach first delegates the question understanding to KG-independent AMR parsing.

Modular Approaches. Frankenstein (Singh et al., 2018) is a system that emphasize the aspect of reusuability where the system learns weights for each reusuable component conditioned on the questions. They neither focus on any KG-independent parsing (AMR) not their results are comparable to any state of the art approaches. (Liang et al., 2021) propose a modular approach for KBQA that uses

an ensemble of phrase mapping techniques and a TreeLSTM-based model for ranking query candidates which requires task specific training data.

5 Discussion

The use of semantic parses such as AMR compared to syntactic dependency parses provides a number of advantages for KBQA systems. First, independent advances in AMR parsing that serve many other purposes can improve the overall performance of the system. For example, on LC-QUAD-1 dev set, a 1.4% performance improvement in AMR Smatch improved the overall system's performance by 1.2%. Recent work also introduces multilingual and domain-specific (biomedical) AMR parsers, which expands the possible domains of application for this work. Second, AMR provides a normalized form of input questions that makes NSQA resilient to subtle changes in input questions with the same meaning. Finally, AMR also transparently handles complex sentence structures such as multi-hop questions or imperative statements.

Nevertheless, the use of AMR semantic parses in NSQA comes with its own set of challenges: 1) Error propagation: Although AMR parsers are very performant (state-of-the-art model achieves an Smatch of over 84%), inter-annotator agreement is only 83% on newswire sentences, as noted in (Banarescu et al., 2013). Accordingly, AMR errors can propagate in NSQA's pipeline and cause errors in generating the correct answer, 2) Granularity mismatch: our proposed path-based AMR transformation is generic and not driven by any domain-specific motivation, but additional adjustments to the algorithm may be needed in new domains due to the different granularity between AMR and SPARQL 3) Optimization mismatch: Smatch, the optimization objective for AMR training, is sub-optimal for KBQA. NSQA requires a particular subset of paths to be correctly extracted, whereas the standard AMR metric Smatch focuses equally on all edge-node triples. We are therefore exploring alternative metrics and how to incorporate them into model training.

6 Conclusion and Future Work

To the best of our knowledge, NSQA is the first system that successfully harnesses a generic semantic parser, particularly AMR, for a KBQA task. Our path-based approach to map AMR to the underlying KG such as DBpedia is first of its kind with promising results in handling compositional queries. NSQA is a modular system where each modules are trained separately for its own task, hence not requiring end-to-end KBQA training. In future, we will explore the potential of the more expressive intermediate logic form with the neuro-symbolic reasoner for KBQA. Particularly, we intend to focus on extending NSQA for temporal reasoning and making it robust to handle incompleteness and inconsistencies in knowledge bases.

References

Ibrahim Abdelaziz, Srinivas Ravishankar, Pavan Kapanipathi, Salim Roukos, and Alexander Gray. 2021. A semantic parsing and reasoning-based approach to knowledge base question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15985–15987.

Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based parsing with stack-transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, page 1001–1007.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 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.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Proceedings of AAAI*.

Nikita Bhutani, Xinyi Zheng, and H V Jagadish. 2019. Learning to answer complex questions over knowledge bases with query composition. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 739–748, New York, NY, USA. Association for Computing Machinery.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752.

Yongrui Chen, Huiying Li, Yuncheng Hua, and Guilin Qi. 2020. Formal query building with query structure prediction for complex question answering over knowledge base. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

William W Cohen, Haitian Sun, R Alex Hofer, and Matthew Siegler. 2020. Scalable neural methods for reasoning with a symbolic knowledge base. *arXiv preprint arXiv:2002.06115*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Dennis Diefenbach, Andreas Both, Kamal Singh, and Pierre Maret. 2020. Towards a question answering system over the semantic web. *Semantic Web*, (Preprint):1–19.

Dennis Diefenbach, Kamal Singh, and Pierre Maret. 2017. Wdaqua-core0: A question answering component for the research community. In *Semantic Web Evaluation Challenge*, pages 84–89. Springer.

Bonnie Dorr, Nizar Habash, and David Traum. 1998. A thematic hierarchy for efficient generation from lexical-conceptual structure. In *Conference of the Association for Machine Translation in the Americas*, pages 333–343. Springer.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *The Semantic Web - ISWC 2019*.

Paul R Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*, pages 1989–1993. Citeseer.

Young-Suk Lee, Ramón Fernandez Astuillo, Tahira Naseem, Revanth Reddy, Radu Florian, and Salim Roukos. 2020. Pushing the limits of amr parsing with self-learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, page 3208–3214.

Shiqi Liang, Kurt Stockinger, Tarcisio Mendes de Farias, Maria Anisimova, and Manuel Gil. 2021. Querying knowledge graphs in natural language. *Journal of Big Data*, 8(1):1–23.

Gaurav Maheshwari, Priyansh Trivedi, Denis Lukovnikov, Nilesh Chakraborty, Asja Fischer, and Jens Lehmann. 2019. Learning to rank query graphs for complex question answering over knowledge graphs. In *International semantic web conference*, pages 487–504. Springer.

Giuseppe M Mazzeo and Carlo Zaniolo. 2016. Answering controlled natural language questions on rdf knowledge bases. In *EDBT*, pages 608–611.

Nandana Mihindukulasooriya, Gaetano Rossiello, Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Mo Yu, Alfio Gliozzo, Salim Roukos, and Alexander Gray. 2020. Leveraging Semantic Parsing for Relation Linking over Knowledge Bases. In *Proceedings of the 19th International Semantic Web Conference (ISWC2020)*.

Tahira Naseem, Srinivas Ravishankar, Nandana Mihindukulasooriya, Ibrahim Abdelaziz, Young-Suk Lee, Pavan Kapanipathi, Salim Roukos, Alfio Gliozzo, and Alexander Gray. 2021. A semantics-aware transformer model of relation linking for knowledge base question answering. In *ACL 2021*.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding smatch: Transition-based amr parsing with reinforcement learning. *arXiv preprint arXiv:1905.13370*.

Hongyu Ren and Jure Leskovec. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs. *Advances in Neural Information Processing Systems*, 33.

Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh Srivastava. 2020. Logical neural networks. *arXiv preprint arXiv:2006.13155*.

Ahmad Sakor, Isaiah Onando Mulang', Kuldeep Singh, Saeedeh Shekarpour, Maria Esther Vidal, Jens Lehmann, and Sören Auer. 2019. Old is gold: Linguistic driven approach for entity and relation linking of short text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2336–2346, Minneapolis, Minnesota. Association for Computational Linguistics.

Kuldeep Singh, Andreas Both, Arun Sethupat, and Saeedeh Shekarpour. 2018. Frankenstein: a platform enabling reuse of question answering components. In *European Semantic Web Conference*, pages 624–638. Springer.

Haitian Sun, Andrew O Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W Cohen. 2020. Faithful embeddings for knowledge base queries.

Advances in Neural Information Processing Systems, 33.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. Lc-quad: A corpus for complex question answering over knowledge graphs. In *ISWC 2017*, pages 210–218.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. 2017. 7th open challenge on question answering over linked data (qald-7). In *Semantic Web Evaluation Challenge*, pages 59–69. Springer.

Svitlana Vakulenko, Javier David Fernandez Garcia, Axel Polleres, Maarten de Rijke, and Michael Cochez. 2019. Message passing for complex question answering over knowledge graphs. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1431–1440.

Dekun Wu, Nana Nosirova, Hui Jiang, and Mingbin Xu. 2019a. A general fofe-net framework for simple and effective question answering over knowledge bases. *arXiv preprint arXiv:1903.12356*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2019b. Zero-shot entity linking with dense entity retrieval. *ArXiv*, abs/1911.03814.

Kun Xu, Yuxuan Lai, Yansong Feng, and Zhiguo Wang. 2019. Enhancing key-value memory neural networks for knowledge based question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2937–2947.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 918–924.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 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)*, pages 571–581.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687.

Weiguo Zheng and Mei Zhang. 2019. Question answering over knowledge graphs via structural query patterns. *arXiv preprint arXiv:1910.09760*.

Lei Zou, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over rdf: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324.