Knowledge Informed Semantic Parsing for Conversational Question Answering

Raghuveer Thirukovalluru[1*], **Mukund Sridhar**[2*],
Dung Thai[1*], **Shruti Chanumolu**[1], **Nicholas Monath**[1],
Shankar Ananthakrishnan[2], **Andrew McCallum**[1]
[1]UMass Amherst, [2]Amazon Alexa AI
{rthirukovall,dthai,schanumolu,nmonath,mccallum}@cs.umass.edu
{harakere,sanantha}@amazon.com

Abstract

Smart assistants are tasked to answer various questions regarding world knowledge. These questions range from retrieval of simple facts to retrieval of complex, multi-hops question followed by various operators (i.e., `filter`, `argmax`). Semantic parsing has emerged as the state-of-the-art for answering these kinds of questions by forming queries to extract information from knowledge bases (KBs). Specially, neural semantic parsers (NSPs) effectively translate natural questions to logical forms, which execute on KB and give desirable answers. Yet, NSPs suffer from non-executable logical forms for some instances in the generated logical forms might be missing due to the incompleteness of KBs. Intuitively, knowing the KB structure informs NSP with changes of the global logical forms structures with respect to changes in KB instances. In this work, we propose a novel knowledge-informed decoder variant of NSP. We consider the conversational question answering settings, where a natural language query, its context and its final answers are available at training. Experimental results show that our method outperformed strong baselines by 1.8 F1 points overall across 10 types of questions of the CSQA dataset. Especially for the "Logical Reasoning" category, our model improves by 7 F1 points. Furthermore, our results are achieved with 90.3% fewer parameters, allowing faster training for large-scale datasets.

1 Introduction

Knowledge base question answering (KBQA) has emerged as an important research topic over the past few years (Sun et al., 2018; Chakraborty et al., 2019; Sun et al., 2019; Shen et al., 2019) alongside with question answering over text corpora. In KBQA, world knowledge is given in the form of multi-relational graph databases (Vrandečić and Krötzsch, 2014; Lehmann et al., 2015) with millions of entities and interrelations between them. When a natural language question arrives, KBQA systems analyse relevant facts in the knowledge bases and derive the answers. In the presence of knowledge bases, question answering results are often time more interpretable and modifiable. For example, the question *"Who started his career at Manchester United in 1992?"* can be answered by fact triples such as (*"David Beckham"*, `member_of_sports_team`, *"Manchester United"*). This fact can be updated as the world knowledge changes while it might be non-trivial to achieve the same effect on text corpora. Likewise, KBQA systems face their own challenges (Chakraborty et al., 2019), especially in the real-world, conversational settings.

In real-world settings, KBQA systems need to perform multi-hop reasoning over chains of supporting facts and carry out various operations within the context of a conversation. For instance, answering the follow up question *"When did he win his first championship?"* might require identifying the player previously mentioned, all of his sport teams, the dates the sport teams won their championships. Then, `argmax` and `filter` operators are applied on the returned dates, yielding answers, i.e., *"1999"* for *"David Beckham"*. Semantic parsing provides a weak supervision framework to learn to perform all these reasoning steps from just the question answer pairs. Semantic parsers define a set of rules (or grammar) for generating logical forms from natural language questions. Candidate logical forms are executable queries on the knowledge bases that yield the corresponding answers. Neural semantic parsers (NSPs) (Liang et al., 2016; Guo et al., 2018; Shen et al., 2019; Guo et al., 2019) employ a neural network to translate natural language questions into logical forms. NSPs have shown good performance on KBQA tasks (Liang et al.,

[*] Equal contribution

2016; Plepi et al., 2021) and further improved with reinforcement learning (Guo et al., 2018), multi-task learning (Shen et al., 2019), and most recently meta-learning (Hua et al., 2020). Most previous works place more emphasis on modeling the reasoning behavior given in the questions than on interactions with the KB. In this work, we propose a KB-aware NSP variant (KISP) to fill in this gap.

One of the main challenges in learning KBQA systems is to adapt to structural changes of the relevant sub-knowledge base. Different reasoning behaviors might apply to similar questions with respect to different sub-knowledge bases. For example, a similar question *"When did Tiger Woods win his first championship?"* would require a different reasoning chain since he didn't participate in a sports team. Structural changes of the sub-KB is a common phenomenon due to the incompleteness nature of knowledge bases. In such cases, knowing the attributes and relations would inform NSPs with changes in logical forms with respect to specific relevant KB entities. To address this problem, we propose a NSPs with a KB-informed decoder that utilizes local knowledge base structure encoded in pre-trained KB embeddings. Our model collects all relevant KB artifacts and integrates their embeddings into each decoding step, iteratively. We also introduce an attention layer on a set of associated KB random walks as an k-steps look ahead that prevents the decoder from going into KB regions where generated logical forms are not executable.

Pre-trained KB embeddings were shown to improve multi-hop KBQA where answers are entities and no operations are involved (Saxena et al., 2020). In this paper, we demonstrate our work on the full KBQA settings with 10 question categories with no constraints on the answers (Saha et al., 2018). While (Saxena et al., 2020) evaluates 2-hop questions (Yih et al., 2016) and 2 and 3-hop questions with limited relation types (Zhang et al., 2018). Our model is also the first NSP variant that utilizes pre-trained features for logical forms generation. CARTON (Plepi et al., 2021) uses an updated action grammar with stacked pointer networks. LASAGNE (Kacupaj et al., 2021) is an extension of CARTON which further includes a graph attention network to exploit correlations between entities, predicates. Empirical results showed that our model improves upon the MaSP model (Shen et al., 2019), a strong baseline for CSQA dataset, by an absolute 1.8 F1, 1.5% accuracy two sets of questions respectively.

Further, we find that by incorporating knowledge-graph information we can match the performance of much larger pre-trained encoder models while using 90.3% fewer parameters.

2 Background

We first formally describe our task and the Neural Semantic Parser (NSP) on which our work is based.

Knowledge Graph: Let $\mathcal{E} = \{e_0 ... e_N\}$ be a set of given entities, and let $\mathcal{R} = \{r_0 ... r_M\}$ be a set of relations. A knowledge graph $\mathcal{G}$ is a set of fact triples in $\mathcal{E} \times \mathcal{R} \times \mathcal{E}$. A triple is represented as (h, r, t) where $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. There is an extensive literature on representing the knowledge graph (Ji et al., 2020; Dai et al., 2020) that encode its semantics and structures. In this work, we use the pre-trained knowledge graph embeddings from Pytorch-BigGraph (Lerer et al., 2019).

Conversational Question Answering: In conversational question answering (CQA), the goal is to answer a question q within the context of the conversation history $\mathcal{C}$. The question q and the history $\mathcal{C}$ are usually concatenated for handling ellipsis and coreference, forming the input $\mathbf{X}$ as $[\mathcal{C}; q]$. At training time, a set of answering entities $\mathcal{A}$ is also given. The set $\mathcal{A}$ comprises entities that resolve to the answer depending on the answer's type. For example, answers of "Simple Question" are a list of entities, the answer of "Verification Question" is Yes/No, whether the set $\mathcal{A}$ is empty or not.

2.1 Neural Semantic Parser

Semantic parsing approach for CQA produces the answer set $\mathcal{A}$ by first generating a logical form $\mathbf{Y}$. Formally, a logical form $\mathbf{Y}$ is a sequence of actions $(y_1, y_2, ..., y_n)$ where the arguments of these actions can be constants (i.e., numbers, dates) or KG instances (i.e., entities, relations, types). The set of actions is defined by a grammar $\mathcal{S}$ (Shen et al., 2019). We consider the weak-supervision settings where the ground truth logical form $\mathbf{Y}$ is not available. Instead, we generate candidates for $\mathbf{Y}$ by performing BFS based on grammar $\mathcal{S}$ over the knowledge graph $\mathcal{G}$ and keeping the candidate logical forms that yield the answer set $\mathcal{A}$ (Guo et al., 2018). Given the input $\mathbf{X}$ and the labeled logical form $\mathbf{Y}$, we train an encoder-decoder neural network to generate logical forms given the question and its conversational context.

Encoder: The input $\mathbf{X}$ is formatted with BERT style. Then, it is fed into a Transformer-based encoder network $\texttt{ENC}$, producing a sequence of encoded states $\mathbf{H} = \texttt{ENC}(\mathbf{X}) = (h_{[CLS]}, h_0, ...)$.

Decoder: The decoder is a Transformer-based model with attention. It takes the input representation from the encoder $h_{[CLS]}$ and the previous decoding state s_{i-1} to produce the target action y_i.

$$Pr_{\mathbf{Y}\sim\mathcal{S}}(\mathbf{Y} \mid \mathbf{X}) = \prod_{y_i \in \mathcal{S}} Pr(y_i \mid s_{i-1}, \mathbf{H}) \quad (1)$$
$$Pr(y_i \mid s_{i-1}, \mathbf{H}) = \text{softmax}(\textsc{Attn}([s_{i-1}; h_{[CLS]}], \mathbf{H}))$$

Classifiers: The decoder is accompanied by a set of classifiers that predict the arguments for the decoder's actions at each decoding step. Our base NSP (Shen et al., 2019) employs FFNNs for relations and entity types classifiers; and pointer networks for entities and constants mentioned in the question. At each decoding step, these classifiers produce an entity e_i, an entity type t_i, a relation r_i, and a constant c_i. The logical form action at time step i is a tuple consists of y_i and its arguments within $\{e_i, t_i, r_i, c_i\}$ defined by the grammar $\mathcal{S}$.

3 Knowledge-Informed Decoder

In this section, we introduce a knowledge-informed decoder that utilizes KG information to generate logical forms. We propose a knowledge injection layer that incorporates KG embeddings into the decoder state at each decoding step. To further inform the decoder with information about the expected structure of the KG, we propose an attention layer on random, k-hops knowledge walks from entities we encounter at each decoding step.

3.1 Knowledge Injection Layer(KIL)

NSP decoders only look at the encoded question and the previous state of decoding to decide the next action. Information of the KB instances (i.e., entities, types, or relations) being considered so far could improve this decision making process. Therefore, at the decoding step i where the action involves a KB instance, we propose a **Knowledge Injection Layer (KIL)** to propagate KB information to the sub-sequence steps. KIL takes in the KB classifiers predictions, incorporates their embeddings into the current encoding state and forwards it to the next decoding step. Eq. 1 becomes

$$Pr_{\mathbf{Y}\sim\mathcal{S}}(\mathbf{Y} \mid \mathbf{X}) = \prod_{y_i \in \mathcal{S}} Pr(y_i \mid s^*_{i-1}, \mathbf{H}) \quad (2)$$
$$s^*_{i-1} = \textsc{Kil}(s_{i-1}) = \text{FNN}([s_{i-1}; \textsc{Embb}(v_{i-1})])$$

where v_{i-1} is the corresponding argument of y_{i-1} and $v_{i-1} \in \mathcal{E} \cup \mathcal{R}$, i.e., $v_i \in \{e_i, t_i, r_i, c_i\}$.

At step j where $j > i$, the decoder is informed of preceding KB instances, and is able to adapt to specific sub-KB structure. We find in cases where there multiple entities in context, having the right entity embedding at timestep j helps logical form in the upcoming steps. The entity embedding carries information about type of the entity, which our model is able to use more appropriate predicates for ambiguous mentions. We empirically show that KIL improves the exact match accuracy of the logical form attributes (logical form without KB).

3.2 Attention on KG Walks (AKW)

Now that the decoder is aware of the previous KB instances, it is also useful to peek at the possible reasoning chains coming out of the current decoding state. We do this to avoid reasoning paths that lead to an non-executable region where the logical form is invalid with respect to the KB. Therefore, we propose an attention look-ahead layer to inspect the upcoming KB structures before making the action prediction. We first generate a set of random walks on the KG from predicted entities and relations with the current decoding step. We then apply the attention look-ahead layer on these KG walks to obtain a representation of the expected KG structures. This representation is then fed back to the decoder to predict the action.

$$Pr_{\mathbf{Y}\sim\mathcal{S}}(\mathbf{Y} \mid \mathbf{X}) = \prod_{y_i \in \mathcal{S}} Pr(y_i \mid s^*_{i-1}, \mathbf{H}, \textsc{RandWalk}(v))$$
$$\textsc{RandWalk}(v) = \textsc{Attn}(\{\textsc{Embb}(p_j \sim \mathcal{G}(v))\}_{j=0..k})$$

where v is one among entities in the question and p_j is a random walk path on the KB starting from v, denoted as $\mathcal{G}(v)$. Here we use one hop random walks from predicates found in the input, though any type of random walk could be used.

With the two proposed layers, our NSP decoder is now fully informed with the past and the future KB structures. We demonstrate that our decoder variant achieves better performance on various question categories. Furthermore, we show that the pre-trained KG embeddings do a significant heavy lifting on representing KB information within the decoder states, resulting in less model parameters and required training data.

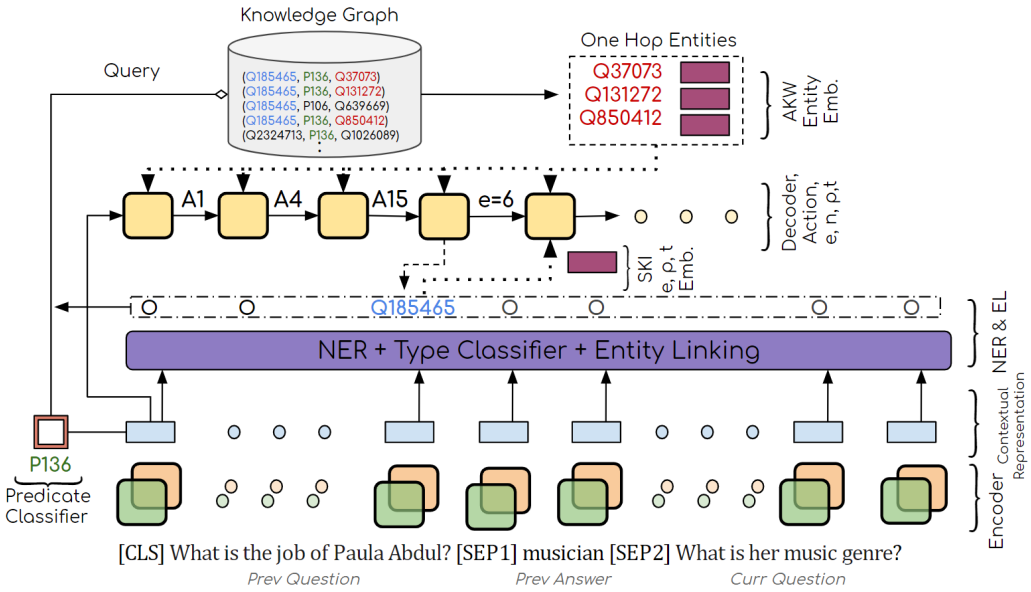

Figure 1: Overall Architecture and the different sources of knowledge used in KISP.

	Methods w\BERT	MaSP	CTN	KISP ♦	KISP ◇
	# train param	155M		157M	160M
F1	Overall	81.20	81.35	82.56	**83.01**
	Clarification	**80.10**	47.31	76.29	76.33
	Comparative	**68.19**	62.0	68.15	67.83
	Logical	76.40	80.80	87.41	**87.14**
	Quantitative	77.31	**80.62**	77.76	77.52
	Simple (Coref.)	78.33	**87.09**	78.78	79.66
	Simple (Direct)	86.57	85.92	87.03	**87.68**
	Simple (Ellipsis)	85.57	85.07	85.86	**86.06**
Acc.	Overall	44.73	**61.28**	46.22	46.22
	Compart.(Count)	28.71	**38.31**	27.65	27.32
	Quant.(Count)	50.07	**57.04**	50.82	50.92
	Verification(Bool)	65.00	**77.82**	72.29	72.72

Table 1: **CSQA w/ Large Models**. CARTON is CTN, KISP(KIL) is KISP♦, KISP(KIL+AKW) is KISP◇.

4 Experiments

Dataset and Evaluation We evaluate our approach on Complex Sequential Question Answering (CSQA) dataset. CSQA consists of 1.6M question answer pairs spread across 200K dialogues. Its has a 152K/16K/28K train, val, test split. More details on the dataset and evaluation metrics used are presented in Section A.1 of the Appendix.

4.1 Main Results

Our model[1] outperforms the MaSP model by 1.8 absolute points in F1-score for entity answer questions and 1.5 absolute points in accuracy for the boolean/counting categories. KISP shows significant improvements in Table 1 compared to MaSP. In more complex question types such 'Logical Reasoning', 'Verification' which require to reason over multiple tuples in the KG and questions that requiring operations like counting, our model outperforms the baseline by more than 10% points. Table 1 compares with MaSP (Shen et al., 2019). Appendix has additional analysis. Our model also beats CARTON (Plepi et al., 2021) in the entity answer questions despite them using an updated action grammar. For boolean, count type questions, the additional action vocabulary helps CARTON out perform our system. We will extend KISP to use this additional action vocabulary in the future.

4.2 Ablation Study

KG informed decoding with small models. A significant performance gain is expected in the smaller models by use of the knowledge graph information. We test this hypothesis by drastically reducing the size of the KISP encoder. This small version of KISP with only 9.7% of the baseline parameter slightly outperforms the baseline BERT model on overall F1-score. The gain comes from the fact that our models receive significant signal from KIL to make a more informed decision of valid actions/types in the next step even without a lot of knowledge from the encoder attention.

Low resource settings. A semantic parsing system as described above typically requires annotated

[1]Code: https://github.com/raghavlite/kisp

	Methods	MaSP (Small)	MaSP (BERT)	KISP◇ (Small)
	# of Parameters	15M	154.8M	15M
F1	Overall	78.91	81.20	**81.52**
	Clarification	75.05	80.10	**82.01**
	Comparative	66.85	68.19	**69.64**
	Logical	69.55	76.40	**84.54**
	Quantitative	74.29	**77.31**	73.9
	Simple (Coref.)	76.27	**78.33**	77.99
	Simple (Direct)	85.39	**86.57**	85.49
	Simple (Ellipsis)	83.04	**85.57**	83.60
Acc.	Overall	38.56	**44.73**	42.55
	Comparative(Count)	22.66	**28.71**	23.65
	Quantitative(Count)	42.73	**50.07**	45.65
	Verification	60.54	65.00	**71.63**

Table 2: Comparison of KISP◇=KISP(KIL+AKW)-Small with different sized baseline models.

golden logical forms for training. Logical form annotation is an resource intensive process (Berant et al., 2013; Liang et al., 2013; Zhong et al., 2017). It is also a difficult process to use brute force computation to find these logical forms; also this process often results in spurious logical forms Shen et al. (2019).

This calls for models which can work with very few training examples. Hence we evaluate the effectiveness of KISP in low resource settings where only a fraction of data is used for training. Table 3 shows that KISP is able to outperform MaSP in these data constrained cases. The gap between MaSP and KISP widens in these low resource settings further justifying our model.

Methods	10% Data		50% Data	
	F1	Acc.	F1	Acc.
MaSP++ (S)	72.99	35.61	79.31	40.27
KISP ♦ (S)	**75.45**	**37.70**	**80.93**	**42.53**

Table 3: Comparison of small KISP(KIL+AKW) and MaSP models. KISP♦=KISP(KIL+AKW)

Met.\Acc.	Sket.	Ent.	Pred.	Type	Num
MaSP (S)	80.55	87.39	97.11	90.62	96.30
KISP◇ (S)	82.32	95.30	98.83	**90.73**	100
KISP♦ (S)	**83.33**	**95.37**	98.83	90.66	100
MASP (B)	83.63	91.90	97.67	**93.11**	100
KISP◇ (B)	84.47	**96.25**	**99.40**	92.25	100
KISP♦ (B)	**85.92**	95.85	99.25	92.25	100

Table 4: Fine grained metrics. KISP◇=KISP(KIL), KISP♦=KISP(KIL+AKW). (S)-Small, (B)-Bert.

Impact of KIL and AKW To further understand how each classifier on the decoder is benefited from the knowledge graph, we look at the accuracies of these classifiers on the evaluation set. Table 4 displays accuracies of the five classifiers from Eq. 1 around logical form generation of different models.

KISP does as better job at predicting the overall skeleton of the logical form - (all the various non e_i, t_i, r_i, c_i) actions. We observe attending to knowledge graph improves the logical form skeleton up to 2.3 points. As shown in Example 3 and 4 of the Appendix, the $count$, $filter$ actions within the logical form are better predicted by KISP. KIL provides entity-embedding for the entity of interest at current timestep this helps the model pick the right predicates in the following steps in ambiguous cases. Cases requiring reasoning benefit from seeing random walks around entities in context - provided by AKW. These lead to better overall sketch accuracy.

KISP is also better at pointing to correct entity accuracy. Pointing to the right entity can has cascading effects on logical form prediction As shown by numbers in Table 4. KISP does a better job with entity pointer improving by almost 4 points. We attribute this to the KIL sytem of KISP which provides the KG embedding for entity of interest at given time step this helps the decoder's entity pointer mechanism.

Entity Linking Errors We follow Sheang (2019) in using a joint mention, type classifier followed by an inverse index entity linker on the input using the encoder representations. The entity pointer classifier described earlier sections looks at these entities in a sentence and points to one among them. We found that a large amount of errors had arisen from this inverse index. Recent work, (Kacupaj et al., 2021) also points this and uses a better entity linker. Improving this module should significantly add to final performance and hence is a very interesting direction for future work.

5 Conclusion

We introduced a neural semantic parsing decoder that uses additional knowledge graph information for Conversational QA. Results show that KISP can significantly boost performance in complex multi-hop question types like logical reasoning questions. Our method can help improve over strong baseline methods like MaSP. Finally we presented a smaller version of our model that is approx 10x smaller without any performance degradation compared to a system that doesn't use KG informed decoding.

References

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.

Nilesh Chakraborty, Denis Lukovnikov, Gaurav Maheshwari, Priyansh Trivedi, Jens Lehmann, and Asja Fischer. 2019. Introduction to neural network based approaches for question answering over knowledge graphs. *arXiv preprint arXiv:1907.09361*.

Yuanfei Dai, Shiping Wang, Neal N Xiong, and Wenzhong Guo. 2020. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics*, 9(5):750.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *Advances in Neural Information Processing Systems*, pages 2942–2951.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2019. Coupling retrieval and meta-learning for context-dependent semantic parsing. *ACL*.

Yuncheng Hua, Yuan-Fang Li, Gholamreza Haffari, Guilin Qi, and Wei Wu. 2020. Retrieve, program, repeat: Complex knowledge base question answering via alternate meta-learning. *arXiv preprint arXiv:2010.15875*.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. 2020. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*.

Endri Kacupaj, Joan Plepi, Kuldeep Singh, Harsh Thakkar, Jens Lehmann, and Maria Maleshkova. 2021. Conversational question answering over knowledge graphs with transformer and graph attention networks. *arXiv preprint arXiv:2104.01569*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. Pytorch-biggraph: A large-scale graph embedding system. *arXiv preprint arXiv:1903.12287*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Joan Plepi, Endri Kacupaj, Kuldeep Singh, Harsh Thakkar, and Jens Lehmann. 2021. Context transformer with stacked pointer networks for conversational question answering over knowledge graphs. *arXiv preprint arXiv:2103.07766*.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Kim Cheng Sheang. 2019. Multilingual complex word identification: Convolutional neural networks with morphological and linguistic features. In *Proceedings of the Student Research Workshop Associated with RANLP 2019*, pages 83–89, Varna, Bulgaria. INCOMA Ltd.

Tao Shen, Xiubo Geng, Tao Qin, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. 2019. Multi-task learning for conversational question answering over a large-scale knowledge base. *EMNLP-IJCNLP*.

Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 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)*, pages 201–206.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Appendices

A.1 Dataset and Evaluation

We evaluate our approach on Complex Sequential Question Answering (CSQA) dataset. CSQA consists of 1.6M question answer pairs spread across 200K dialogues. Its has a 152K/16K/28K train, val, test split. The dataset's knowledge graph is built on wikidata (Vrandečić and Krötzsch, 2014) and represented with triples. The KB consists of 21.2M triplets over 12.8M entities, 3054 distinct entity types, and 567 distinct predicates. There are 10 different question categories split into two groups. Answers to the first group of questions are a list of entities. Question categories of this group are evaluated by the macro F1 score between predicted entities and golden entities. Answers to question categories in the second group are either counts or boolean. This group is evaluated by accuracy. Overall scores for each group are the weighted averaged metrics of all the categories in the group. We refer the reader to Saha et al. (2018) for a more detailed understanding of different categories of questions. Following sections contain training/eval specifics.

A.2 Training details & Evaluation Metrics

We followed Shen et al (2019) to search for logical forms and create the training data. Exact hyperparameters used in the experiments are mentioned below. We followed Saha et al. (2018) for evaluation metrics. Macro Precision and Macro Recall were used when the answer was a list of entities. For questions with answer type boolean/number, we use accuracy.

A.3 Training time Analysis

Training times of different models are in Table.5

Model	Training Time (hrs)
MaSP++	6
KISP(SKI)	7.5
KISP(SKI+AKW)	8
MASP++ (BERT)	27.4
KISP(SKI) BERT	29.5
KISP(SKI + AKW) BERT	32
KISP(SKI + AKW) small	4.5
MASP++ small	3

Table 5: Running times of different models

There are some known in-efficiencies in the code, some from design and others conceptual. We intend to improve training time in future work by incorporating more e2e methods that will reduce GPU2CPU & CPU2GPU communication and also through some design changes in the short term.

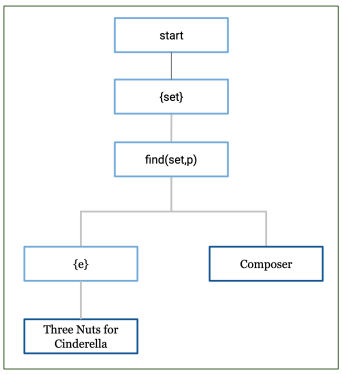

Figure 2: Example1 logical form KISP(SKI+AKW)BERT

A.4 Logical form Analysis

We identify examples to show performance improvement in KISP models, in predicting the correct answer and logical form. As shown in Table 6 below, KISP models for these examples do a better job at sketch, entity, num, type, predicate classification compared to MaSP. The coloured images in Figure 2- 6 show the differences between MaSP and KISP models. For each example we show the golden logical form tree(also predicted by one of the KISP models), MaSP's logical form and the mistakes made by the baseline in color red.

Example1

- **Utterance**
  - Q: Which works of art stars Jiří Růžička as actor and originated in Germany ?
  - A: Three Nuts for Cinderella
  - Q: Who was that work of art composed by ?
  - A: Karel Svoboda
- **Logical form**
  - @ Gold
    find({Three Nuts for Cinderella}, Composer)

Example2

- **Utterance**
  - Q: What is the job of Joe Falcon ?
  - A: musician
  - Q: What can be considered as category for Joe Falcon ?

Example	Curr_Question type	Predicted logical form = Gold logical form and Predicted answer = Gold answer			
		MaSP	KISP(SKI)	KISP(SKI+AKW)	KISP(SKI+AKW): BERT
Example1	Simple Question (Coreferenced)	Yes	Yes	Yes	Yes
Example2	Simple Question (Direct)	No	Yes	Yes	Yes
Example3	Quantitative Reasoning (Count) (All)	No	Yes	Yes	Yes
Example4	Quantitative Reasoning (Count) (All)	No	No	Yes	Yes
Example5	Logical Reasoning (All)	No	No	No	Yes

Table 6: Examples are based on the predicted logical form and answers in comparison to their gold counterpart

A: Cajun music

- **Logical form**

@ Gold
find({Joe Falcon}, genre)

@ MaSP
find({Joe Falcon}, Occupation)

Example3

- **Utterance**

Q: How many administrative territories have atleast 4 administrative territories or french administrative divisions as their capital?

A: 1

Q: How many cities are associated to Albania as the capital?

A: 1

- **Logical form**

@ Gold
count({*find*({Albania}, capital)})

@ MaSP
count({*filter*(city,{*union*({*find*({Albania},capital)},{Albania})})})

Example4

- **Utterance**

Q: Is France the native country of Charles Boyer ?

A: YES

Q: How many works of art stars Charles Boyer as actor ?

A: 68

- **Logical form**

@ Gold
count(*filter*(work of art,{*find*({Charles Boyer}, cast member)}))

@ MaSP
count({*union*({*filter*(work of art,{*find*({France},country of origin)})},{*find*({Charles Boyer},cast member)})})

@ KISP(SKI)
count({*diff*(*argmax*({*filter*(work of art, {Charles Boyer})},cast member),{*filter*(work of art,{Charles Boyer})})})

Example5

- **Utterance**

Q: What is that person a member of ?

A: Tunisia national football team

Q: Which recurring events did Tunisia national football team and Alberto García Aspe participate in ?

A: 2002 FIFA World Cup, 1998 FIFA World Cup

- **Logical form**

@ Gold
inter({*find*({Tunisia national football team}, participant)},{*find*({Alberto García Aspe}, participant)})

@ MaSP
inter(*find*({Alberto García Aspe}, participant), *find*({Alberto García Aspe}, participant))

@ KISP(SKI)
filter(recurring event,{*union*({*find*({Tunisia national football team}, participant)},{Alberto García })

@ KISP(SKI+AKW)
inter({*find*({Tunisia national football team }, participant)},{*find*({Tunisia national football team }, participant)})

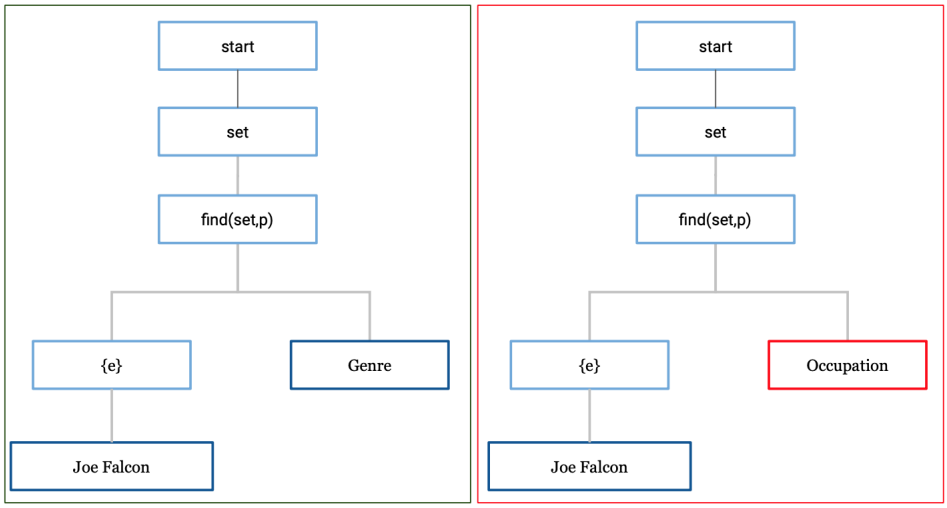

Figure 3: Example2 logical form KISP(SKI) vs MaSP

Example2 (Figure 3) and Example5 (Figure 6) show improvement in logical form entity and predicate instantiation compared to MaSP. We notice improvement in logical form skeleton for KISP(SKI+AKW)BERT model in Example4 (Figure 5). Example 3 (Figure 4) is a case where MaSP gets the right answer despite the incorrect logical form.

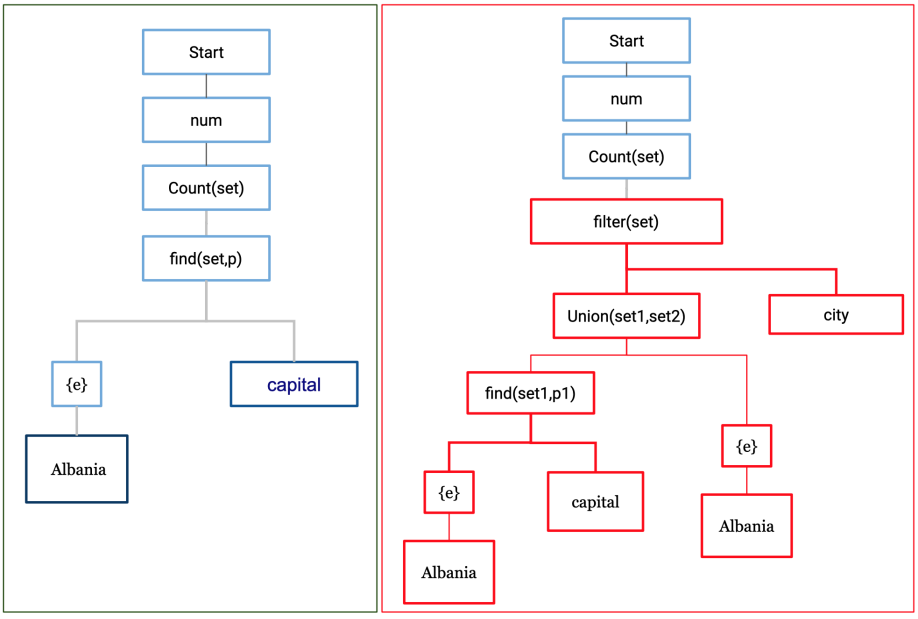

Figure 4: Example3 logical form KISP(SKI+AKW) vs MaSP

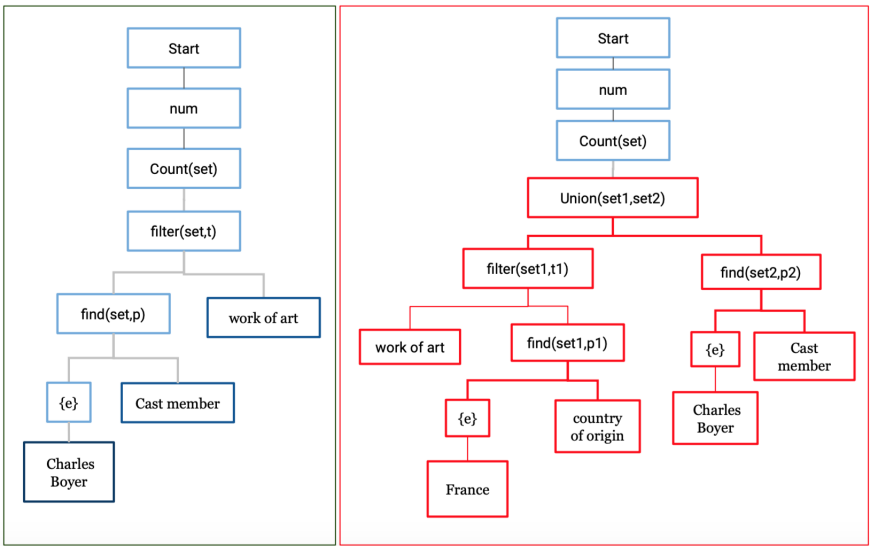

Figure 5: Example4 logical form KISP(SKI+AKW) vs MaSP

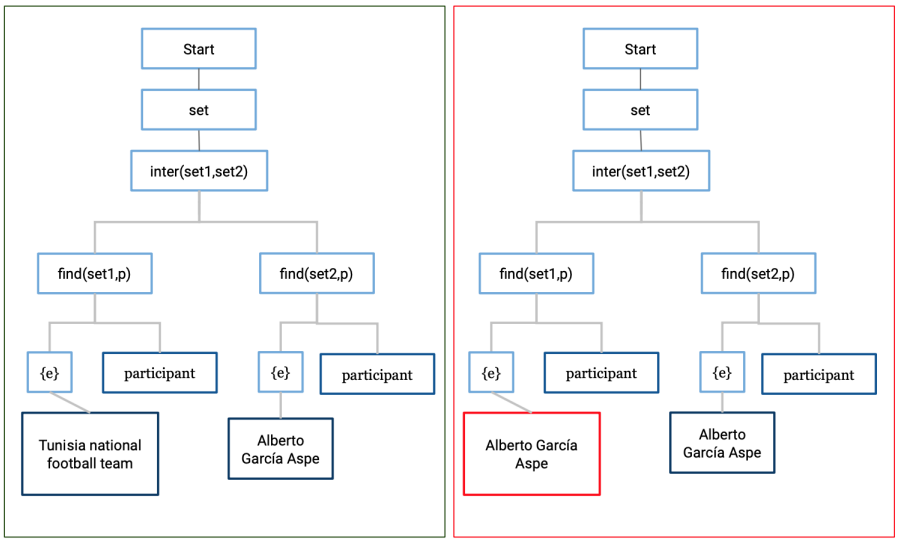

Figure 6: Example5 logical form KISP(SKI+AKW)BERT vs MaSP