Case-Based Reasoning for Natural Language Queries over Knowledge Bases

Rajarshi Das[1], **Manzil Zaheer**[3], **Dung Thai**[1], **Ameya Godbole**[1], **Ethan Perez**[2], **Jay-Yoon Lee**[1], **Lizhen Tan**[4], **Lazaros Polymenakos**[5]*, **Andrew McCallum**[1]
[1]University of Massachusetts, Amherst, [2]New York University
[3]Google Research, [4]Amazon, [5]EY, Greece
{rajarshi,dthai,agodbole,jaylee,mccallum}@cs.umass.edu

Abstract

It is often challenging to solve a complex problem from scratch, but much easier if we can access other similar problems with their solutions — a paradigm known as case-based reasoning (CBR). We propose a neuro-symbolic CBR approach (CBR-KBQA) for question answering over large knowledge bases. CBR-KBQA consists of a nonparametric memory that stores cases (question and logical forms) and a parametric model that can generate a logical form for a new question by retrieving cases that are relevant to it. On several KBQA datasets that contain complex questions, CBR-KBQA achieves competitive performance. For example, on the COMPLEXWEBQUESTIONS dataset, CBR-KBQA outperforms the current state of the art by 11% on accuracy. Furthermore, we show that CBR-KBQA is capable of using new cases *without* any further training: by incorporating a few human-labeled examples in the case memory, CBR-KBQA is able to successfully generate logical forms containing unseen KB entities as well as relations.

1 Introduction

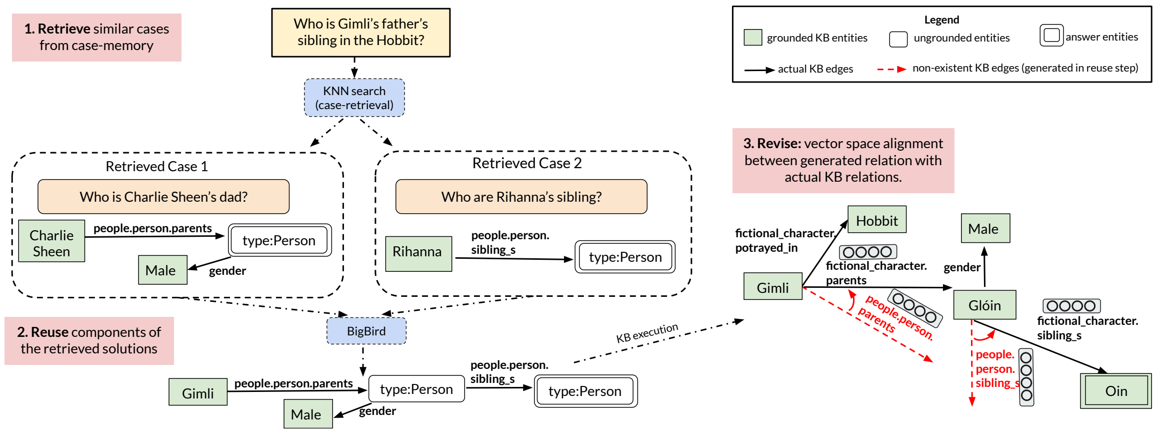
Humans often solve a new problem by recollecting and adapting the solution to multiple related problems that they have encountered in the past (Ross, 1984; Lancaster and Kolodner, 1987; Schmidt et al., 1990). In classical artificial intelligence (AI), case-based reasoning (CBR) pioneered by Schank (1982), tries to incorporate such model of reasoning in AI systems (Kolodner, 1983; Rissland, 1983; Leake, 1996). A sketch of a CBR system (Aamodt and Plaza, 1994) comprises of — (i) a retrieval module, in which 'cases' that are similar to the given problem are retrieved, (ii) a reuse module, where the solutions of the retrieved cases are re-used to synthesize a new solution. Often, the new solution does not work and needs more revision, which is handled by a (iii) revise module.

In its early days, the components of CBR were implemented with symbolic systems, which had their limitations. For example, finding similar cases and synthesizing new solutions from them is a challenging task for a CBR system implemented with symbolic components. However, with the recent advancements in representation learning (LeCun et al., 2015), the performance of ML systems have improved substantially on a range of practical tasks.

Given a query, CBR-KBQA uses a neural retriever to retrieve other similar queries (and their logical forms) from a case memory (e.g. training set). Next, CBR-KBQA generates a logical form for the given query by learning to reuse various components of the logical forms of the retrieved cases. However, often the generated logical form does not produce the right answer when executed against a knowledge base (KB). This can happen because one or more KB relations needed are never present in the retrieved cases or because KBs are woefully incomplete (Min et al., 2013) (Figure 1). To alleviate such cases, CBR-KBQA has an additional revise step that *aligns* the generated relations in the logical form to the query entities' local neighborhood in the KB. To achieve this, we take advantage of pre-trained relation embeddings from KB completion techniques (e.g. Trans-E (Bordes et al., 2013)) that learn the structure of the KB.

It has been shown that neural seq2seq models do not generalize well to novel combinations of previously seen input (Lake and Baroni, 2018; Loula et al., 2018). However, CBR-KBQA has the ability to reuse relations from *multiple* retrieved cases, even if each case contains only partial logic to answer the query. We show that CBR-KBQA is effec-

*Now director of AI at EY CESA, Greece. Work done while at Amazon



1. Retrieve similar cases from case-memory
Who is Gimli's father's sibling in the Hobbit?
KNN search (case-retrieval)
Legend
grounded KB entities
ungrounded entities
answer entities
actual KB edges
non-existent KB edges (generated in reuse step)
Retrieved Case 1
Who is Charlie Sheen's dad?
Charlie Sheen
people.person.parents
type:Person
Male
gender
Retrieved Case 2
Who are Rihanna's sibling?
Rihanna
people.person.sibling_s
type:Person
3. Revise: vector space alignment between generated relation with actual KB relations.
fictional_character.potrayed_in
Hobbit
Male
fictional_character.parents
gender
Gimli
Glóin
people.person.parents
fictional_character.sibling_s
people.person.sibling_s
Oin
2. Reuse components of the retrieved solutions
BigBird
KB execution
Gimli
people.person.parents
type:Person
people.person.sibling_s
type:Person
Male
gender

Figure 1: CBR-KBQA derives the logical form (LF) for a new query from the LFs of other retrieved queries from the case-memory. However, the derived LF might not execute because of missing edges in the KB. The revise step aligns any such missing edges (relations) with existing semantically-similar edges in the KB.

tive for questions that need novel combination of KB relations, achieving competitive results on multiple KBQA benchmarks such as WebQuestionsSP (Yih et al., 2016), ComplexWebQuestions (CWQ) (Talmor and Berant, 2018) and CompositionalFreebaseQuestions (CFQ) (Keysers et al., 2020). For example, on the hidden test-set of the challenging CWQ dataset, CBR-KBQA outperforms the best system by over 11% points.

We further demonstrate that CBR-KBQA, without the need of any further fine-tuning, also generalizes to queries that need relations which were *never seen* in the training set. This is possible due to CBR-KBQA's nonparametric approach which allows one to *inject* relevant simple cases during inference, allowing it to reuse new relations from those cases. In a controlled human-in-the-loop experiment, we show that CBR-KBQA can correctly answer such questions when an expert (e.g. database administrator) injects a few simple cases to the case memory. CBR-KBQA is able to retrieve those examples from the memory and use the unseen relations to compose new logical forms for the given query.

Generalization to unseen KB relations, without any re-training, is out of scope for current neural models. Currently, the popular approach to handle such cases is to re-train or fine-tune the model on new examples. This process is not only time-consuming and laborious but models also suffer from catastrophic forgetting (Hinton and Plaut, 1987; Kirkpatrick et al., 2017), making wrong predictions on examples which it previously predicted correctly. We believe that the controllable properties of CBR-KBQA are essential for QA models to be deployed in real-world settings and hope that our work will inspire further research in this direction.

Recent works such as REALM (Guu et al., 2020) and RAG (Lewis et al., 2020b) retrieve relevant paragraphs from a nonparametric memory for answering questions. CBR-KBQA, in contrast, retrieves *similar queries* w.r.t the input query and uses the relational similarity between their logical forms to derive a logical form for the new query. CBR-KBQA is also similar to the recent retrieve and edit framework (Hashimoto et al., 2018) for generating structured output. However, unlike us they condition on only a single retrieved example and hence is unlikely to be able to handle complex questions that need reuse of partial logic from multiple questions. Moreover, unlike CBR-KBQA, retrieve and edit does not have a component that can explicitly revise an initially generated output.

The contributions of our paper are as follows — (a) We present a neural CBR approach for KBQA capable of generating complex logical forms conditioned on similar retrieved questions and their logical forms. (b) Since CBR-KBQA explicitly learns to reuse cases, we show it is able to generalize to unseen relations at test time, when relevant cases are provided. (c) We also show the efficacy of our revise step of CBR-KBQA which allows to correct generated output by aligning it to local neighborhood of the query entity. (d) Lastly, we show that CBR-KBQA significantly outperforms other competitive models on several KBQA benchmarks.

2 Model

This section describes the implementation of various modules of CBR-KBQA. In CBR, a case is defined as an abstract representation of a problem along with its solution. In our KBQA setting, a case is a natural language query paired with an executable logical form. The practical importance of KBQA has led to the creation of an array of recent datasets (Zelle and Mooney, 1996; Bordes et al., 2015; Su et al., 2016; Yih et al., 2016; Zhong et al., 2017a; Ngomo, 2018; Yu et al., 2018; Talmor and Berant, 2018, inter-alia). In these datasets, a question is paired with an executable logical form such as SPARQL, SQL, S-expression or graph query. All of these forms have equal representational capacity and are interchangeable (Su et al., 2016). Figure 2 shows an example of two equivalent logical forms. For our experiments, we consider SPARQL programs as our logical form.

Formal definition of task: let q be a natural language query and let $\mathcal{K}$ be a symbolic KB that needs to be queried to retrieve an answer list $\mathcal{A}$ containing the answer(s) for q. We also assume access to a training set $\mathcal{D} = \{(q_1, \ell_1), (q_2, \ell_2), \ldots (q_N, \ell_N)\}$ of queries and their corresponding logical forms where q_i, ℓ_i represents query and its corresponding logical form, respectively. A logical form is an executable query containing entities, relations and free variables (Figure 2). CBR-KBQA first retrieves K similar cases $\mathcal{D}_q$ from $\mathcal{D}$ (§ 2.1). It then generates a intermediate logical form ℓ_{inter} by learning to reuse components of the logical forms of the retrieved cases (§ 2.2). Next, the logical form ℓ_{inter} is revised to output the final logical form ℓ by aligning to the relations present in the neighborhood subgraph of the query entity to recover from any spurious relations generated in the reuse step (§ 2.3). Finally, ℓ is executed against $\mathcal{K}$ and the list of answer entities are returned. We evaluate our KBQA system by calculating the accuracy of the retrieved answer list w.r.t a held-out set of queries.

2.1 Retrieve

The retrieval module computes dense representation of the given query and uses it to retrieve other similar query representation from a training set. Inspired by the recent advances in neural dense passage retrieval (Das et al., 2019; Karpukhin et al., 2020), we use a ROBERTA-base encoder to encode each question independently. Also, we want to retrieve questions that have high relational similarity instead of questions which share the same entities (e.g. we prefer to score the query pair (Who is Justin Bieber's brother?, Who is Rihanna's brother?), higher than (Who is Justin Bieber's brother?, Who is Justin Bieber's father?)). To minimize the effect of entities during retrieval, we use a named entity tagger[1] to detect spans of entities and mask them with [BLANK] symbol with a probability p_{mask}, during training. The entity masking strategy has previously been successfully used in learning entity-independent relational representations (Soares et al., 2019). The similarity score between two queries is given by the inner product between their normalized vector representations (cosine similarity), where each representation, following standard practice (Guu et al., 2020), is obtained from the encoding of the initial [CLS] token of the query.

Fine-tuning question retriever: In passage retrieval, training data is gathered via distant supervision in which passages containing the answer is marked as a positive example for training. Since in our setup, we need to retrieve similar questions, we use the available logical forms as a source of distant supervision. Specifically, a question pair is weighed by the amount of overlap (w.r.t KB relations) it has in their corresponding logical queries. Following DPR (Karpukhin et al., 2020), we ensure there is at least one positive example for each query during training and use a weighted negative log-likelihood loss where the weights are computed by the F_1 score between the set of relations present in the corresponding logical forms. Concretely, let $(\text{q}_1, \text{q}_2, \ldots, \text{q}_\text{B})$ denote all questions in a mini-batch. The loss function is:

$$L = -\sum_{i,j} w_{i,j} \log \frac{\exp(\text{sim}(\mathbf{q_i}, \mathbf{q_j}))}{\sum_j \exp(\text{sim}(\mathbf{q_i}, \mathbf{q_j}))}$$

Here, $\mathbf{q_i} \in \mathbb{R}^d$ denotes the vector representation of query q_i and $\text{sim}(\mathbf{q_i}, \mathbf{q_j}) = \mathbf{q_i}^\top \mathbf{q_j}$. $w_{i,j}$ is computed as the F_1 overlap between relations in the logical pairs of q_i and q_j. We pre-compute and cache the query representations of the training set $\mathcal{D}$. For query q, we return the top-k similar queries in $\mathcal{D}$ w.r.t q and pass it to the reuse module.

2.2 Reuse

The reuse step generates an intermediate logical form from the k cases that are fed to it as input from the retriever module. Pre-trained encoder-decoder transformer models such as BART (Lewis et al., 2020a) and T5 (Raffel et al., 2020) have enjoyed dramatic success on semantic parsing (Lin

[1]https://cloud.google.com/natural-language

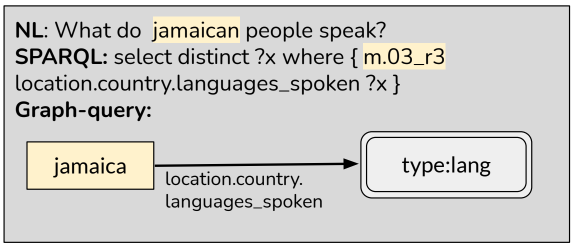

NL: What do jamaican people speak?
SPARQL: select distinct ?x where { m.03_r3 location.country.languages_spoken ?x }
Graph-query:

Figure 2: An example of a SPARQL logical form for a simple query and its equivalent graph-query.

et al., 2018; Hwang et al., 2019; Shaw et al., 2020; Suhr et al., 2020). We take a similar approach in generating an intermediate logical form conditioned on the retrieved cases. However, one of the core limitation of transformer-based models is its quadratic dependency (in terms of memory), because of full-attention, which severely limits the sequence length it can operate on. For example, BART and T5 only supports sequence length of 512 tokens in its encoder. Recall that for us, a case is a query from the train set and an executable SPARQL program, which can be arbitrarily long.

To increase the number of input cases, we leverage a recently proposed sparse-attention transformer architecture — BIGBIRD (Zaheer et al., 2020). Instead of having each token attend to all input tokens as in a standard transformer, each token attends to only nearby tokens. Additionally, a small set of global tokens attend to all tokens in the input. This reduces the transformer's memory complexity from quadratic to linear, and empirically, BIGBIRD enables us to use many more cases.

Description of input: The input query q and cases $\mathcal{D}_q = \{(q'_1, \ell'_1), (q'_2, \ell'_2), \dots (q'_k, \ell'_k)\}$ are concatenated on the encoder side. Specifically, $\mathtt{Input_{ENC}}(\mathtt{q}, \mathcal{D}_\mathtt{q}) = \mathtt{q}\,\mathtt{[SEP]}\,\mathtt{q}'_1\,\mathtt{[SEP]}\,\ell'_1, \dots \mathtt{q_k}'\,\mathtt{[SEP]}\,\ell_\mathtt{k}'$, where `[SEP]` denotes the standard separator token. Each logical form also contain the KB entity id of each entities in the question (e.g. m.03_r3 for Jamaica in Figure 2). We append the entity id after the surface form of the entity mention in the question string. For example, the query in Figure 2 becomes "What do Jamaican m.03_r3 people speak?".

Training is done using a standard seq2seq cross-entropy objective. Large deep neural networks usually benefit from "good" initialization points (Frankle and Carbin, 2019) and being able to utilize pre-trained weights is critical for seq2seq models. We find it helpful to have a regularization term that minimizes the Kullback–Leibler divergence (KLD) between output softmax layers of (1) when only the query q is presented (i.e not using cases), and (2) when query and cases ($\mathcal{D}_q$) are available (Yu et al., 2013). Formally, let f be the seq2seq model, let $\sigma = softmax(f(q, \mathcal{D}_q))$ and $\sigma' = softmax(f(q))$ be the decoder's prediction distribution with and without cases, respectively. The following KLD term is added to the seq2seq cross-entropy loss

$$L = L_{CE}(f(q, \mathcal{D}_q), l_q) + \lambda_T \mathrm{KLD}(\sigma, \sigma')$$

where $\lambda_T \in [0, 1]$ is a hyper-parameter. Intuitively, this term regularizes the prediction of $f(q, \mathcal{D}_q)$ not to deviate too far away from that of the $f(q)$ and we found this to work better than initializing with a model not using cases.

2.3 Revise

In the previous step, the model explicitly reuses the relations present in $\mathcal{D}_q$, nonetheless, there is no guarantee that the query relations in $\mathcal{D}_q$ will contain the relations required to answer the original query q. This can happen when the domain of q and domain of cases in $\mathcal{D}_q$ are different even when the relations are semantically similar. For example, in Figure 1 although the retrieved relations in NN queries are semantically similar, there is a domain mismatch (person v/s fictional characters). Similarly, large KBs are very incomplete (Min et al., 2013), so querying with a valid relation might require an edge that is missing in the KB leading to intermediate logical forms which do not execute.

To alleviate this problem and to make the queries executable, we explicitly *align* the generated relations with relations (edges) present in the local neighborhood of the query entity in the KG. We propose the following alignment models:

Using pre-trained KB embeddings: KB completion is an extensively studied research field (Nickel et al., 2011; Bordes et al., 2013; Socher et al., 2013; Velickovic et al., 2018; Sun et al., 2019b) and several methods have been developed that learn low dimensional representation of relations such that similar relations are closer to each other in the embedding space. We take advantage of the pre-trained relations obtained from TransE (Bordes et al., 2013), a widely used model for KB completion. For each predicted relation, we find the most similar (outgoing or incoming) relation edge (in terms of cosine similarity) that exists in the

Model	P	R	F1	Acc
Weakly supervised models				
GraftNet (Sun et al., 2018)	-	-	66.4†	-
PullNet (Sun et al., 2019a)	-	-	68.1†	-
EmbedKGQA (Saxena et al., 2020)	-	-	66.6†	-
Supervised models				
STAGG (Yih et al., 2016)	70.9	**80.3**	71.7	63.9
T5-11B (Raffel et al., 2020)	62.1	62.6	61.5	56.5
T5-11B + Revise	63.6	64.3	63.0	57.7
CBR-KBQA (Ours)	**73.1**	75.1	**72.8**	**69.9**

Table 1: Performance on the WebQSP dataset. GraftNet, PullNet and EmbedKGQA produces a ranking of KG entities hence evaluation is in Hits@k (see text for description). CBR-KBQA significantly outperforms baseline models in the strict exact match accuracy metric. † Models report hits@1 instead of F1

Dataset	Precision	Recall	F1
WebQSP	0.761	0.819	0.789
CWQ	0.707	0.910	0.796

Table 2: Entity linking performance on various datasets

KB for that entity and align with it. If the predicted edge exists in the KB, it trivially aligns with itself. There can be multiple missing edges that needs alignment (Figure 1) and we find it more effective to do beam-search instead of greedy-matching the most similar edge at each step.

Using similarity in surface forms: Similar relations (even across domains) have overlap in their surface forms (e.g. 'siblings' is common term in both 'person.siblings' and 'fictional_character.siblings'). Therefore, word embeddings obtained by encoding these words should be similar. This observation has been successfully utilized in previous works (Toutanova and Chen, 2015; Hwang et al., 2019). We similarly encode the predicted relation and all the outgoing or incoming edges with ROBERTA-base model. Following standard practices, relation strings are prepended with a [CLS] token and the word pieces are encoded with the ROBERTA-base model and the output embedding of the [CLS] token is considered as the relation representation. Similarity between two relation representations is computed by cosine similarity.

Our alignment is simple and requires no learning. By aligning only to individual edges in the KB, we make sure that we do not change the structure of the generated LF. We leave the exploration of learning to align single edges in the program to sequence of edges (paths) in the KB as future work.

3 Experiments

Data: For all our experiments, the underlying KB is full Freebase containing over 45 million entities (nodes) and 3 billion facts (edges) (Bollacker et al., 2008). We test CBR-KBQA on three datasets — WebQSP (Yih et al., 2016), CWQ (Talmor and Berant, 2018) and CFQ (Keysers et al., 2020). Please refer to §A.1 for a detailed description of each datasets.
Hyperparameters: All hyperparameters are set by tuning on the valdation set for each dataset. We initialize our retriever with the pre-trained ROBERTA-base weights. We set $p_{\text{mask}} = 0.2$ for CWQ and 0.5 for the remaining datasets. We use a BIGBIRD generator network with 6 encoding and 6 decoding sparse-attention layers, which we initialize with pre-trained BART-base weights. We use k=20 cases and decode with a beam size of 5. Initial learning rate is set to 5×10^{-5} and is decayed linearly through training. Further details for the EMNLP reproducibility checklist is given in §A.2.

3.1 Entity Linking

The first step required to generate an executable LF for a NL query is to identify and link the entities present in the query. For our experiments, we use a combination of an off-the-shelf entity linker and a large mapping of mentions to surface forms. For the off-the-shelf linker, we use a recently proposed high precision entity linker ELQ (Li et al., 2020). To further improve recall of our system, we first identify mention spans of entities in the question by tagging it with a NER[2] system. Next, we link entities not linked by ELQ by exact matching with surface form annotated in FACC1 project (Gabrilovich et al., 2013). Our entity linking results are shown in Table 2.

3.2 KBQA Results

This section reports the performance of CBR-KBQA on various benchmarks. We report the strict exact match accuracy where we compare the list of predicted answers by executing the generated SPARQL program to the list of gold answers[3]. A question is answered correctly if the two list match exactly. We also report the precision, recall and the F1 score to be comparable to the baselines. Models such as GraftNet (Sun et al., 2018) and PullNet (Sun et al.,

[2]`https://cloud.google.com/natural-language`

[3]We generate the gold answer entities by executing the gold SPARQL query against our Freebase KB

Model	P	R	F1	Acc
Weakly supervised models				
KBQA-GST (Lan et al., 2019)	-	-	-	39.4
QGG (Lan and Jiang, 2020)	-	-	-	44.1
PullNet (Sun et al., 2019a)	-	-	-	45.9
DynAS (Alibaba Group)	-	-	-	50.0
Supervised models				
T5-11B (Raffel et al., 2020)	55.2	55.4	54.6	52.4
T5-11B + Revise	58.7	59.6	58.2	55.6
CBR-KBQA (Ours)	**70.4**	**71.9**	**70.0**	**67.1**

Table 3: Performance on the hidden test set of CWQ.

Model	MCD1	MCD2	MCD3	MCD-mean
T5-11B	72.9	69.2	62.0	67.7
CBR-KBQA	**87.8**	**75.1**	**71.5**	**78.1**

Table 4: Performance (accuracy) on the CFQ dataset.

2019a) rank answer entities and return the top entity as answer (Hits@1 in table 1). This is undesirable for questions that have multiple entities as answers (e.g. "Name the countries bordering the U.S.?"). We also report performance of models that only depend on the query and answer pair during training and do not depend on LF supervision (weakly-supervised setting). Unsurprisingly, models trained with explicit LF supervision perform better than weakly supervised models. Our main baseline is a massive pre-trained seq2seq model with orders of magnitude more number of parameters — T5-11B (Raffel et al., 2020). T5 has recently been shown to be effective for compositional KBQA (Furrer et al., 2020). For each dataset, we fine-tune the T5 model on the query and the LF pairs.

Table 1 reports results of various models on WebQSP. All reported model except CBR-KBQA and T5-11B directly operate on the KB (e.g. traverse KB paths starting from the query entity) to generate the LF or the answer. As a result, models such as STAGG tend to enjoy much higher recall. On the other hand, much of our logical query is generated by reusing components of similar cases. We also report the results of 'aligning' the LF produced by T5 using our revise step. As shown in Table 1, CBR-KBQA outperforms all other models significantly and improves on the strict exact-match accuracy by more than 6 points w.r.t. the best model. Revise step also improves on the performance of T5 suggesting that it is generally applicable. Table 3 reports performance on the hidden test set of CWQ[4], which was built by extending WebQSP questions with the goal of making a more complex multi-hop questions. It is encouraging to see that CBR-KBQA outperforms all other baselines on this challenging dataset by a significant margin. Finally, we report results on CFQ in Table 4. On error analysis, we found that on several questions which are yes/no type, our model was predicting the list of correct entities instead of predicting a yes or no. We created a rule-based binary classifier that predicts the type of question (yes/no or other). If the question was predicted as a yes/no, we would output a yes if the length of the predicted answer list was greater than zero and no otherwise. (If the model was already predicting a yes/no, we keep the original answer unchanged). We report results on all the three MCD splits of the dataset and compare with the T5-11B model of Furrer et al. (2020) and we find that our model outperforms T5-11B on this dataset as well. It is encouraging to see that CBR-KBQA, even though containing order-of-magnitudes less parameters than T5-11B, outperforms it on all benchmarks showing that it is possible for smaller models with less carbon footprint and added reasoning capabilities to outperform massive pre-trained LMs.

[4]The result of our model in the official leaderboard (`https://www.tau-nlp.org/compwebq-leaderboard`) is higher (70.4 vs 67.1). This is because the official evaluation script assigns full score if any of the correct answer entities are returned even if there are multiple correct answers for a question. In the paper we report strict exact-match accuracy.

3.3 Efficacy of Revise step

Table 5 show that the revise step is useful for CBR-KBQA on multiple datasets. We also show that the T5 model also benefits from the alignment in revise step with more than 3 points improvement in F1 score on the CWQ dataset. We find that TransE alignment outperforms ROBERTA based alignment, suggesting that graph structure information is more useful than surface form similarity for aligning relations. Moreover, relation names are usually short strings, so they do not provide enough context for LMs to form good representations.

Next we demonstrate the advantage of the non-parametric property of CBR-KBQA— ability to fix an initial wrong prediction by allowing new cases to be *injected* to the case-memory. This allows CBR-KBQA to generalize to queries which needs relation *never seen* during training. Due to space constraints, we report other results (e.g. retriever performance), ablations and other analysis in §B.

WebQSP	Accuracy(%)	Δ
CBR-KBQA (before Revise)	69.43	–
+Revise (Roberta)	69.49	+0.06
+Revise (TransE)	**70.00**	**+0.57**
CWQ	Accuracy(%)	Δ
CBR-KBQA (before Revise)	65.95	–
+Revise (Roberta)	66.32	+0.37
+Revise (TransE)	**67.11**	**+1.16**

Table 5: Impacts of the revise step. We show that the revise step consistently improves the accuracy on WebQSP and CWQ, especially with the TransE pretrained embeddings.

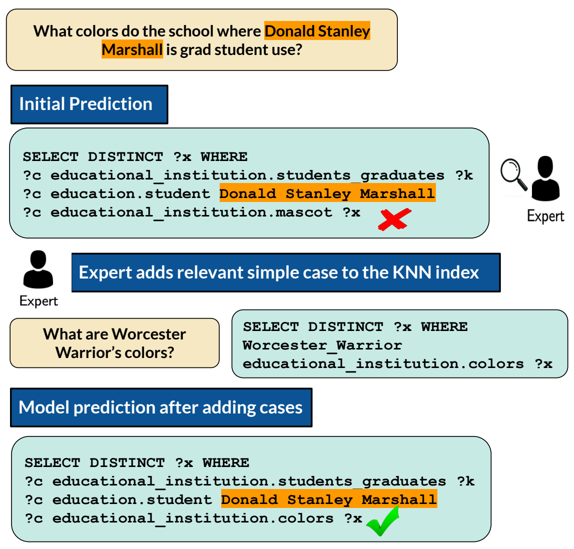
3.4 Point-Fixes to Model Predictions

Modern QA systems built on top of large LMs do not provide us the opportunity to debug an erroneous prediction. The current approach is to fine-tune the model on new data. However, this process is time-consuming and impractical for production settings. Moreover, it has been shown (and as we will empirically demonstrate) that this approach leads to catastrophic forgetting where the model forgets what it had learned before. (McCloskey and Cohen, 1989; Kirkpatrick et al., 2017). On the other hand, CBR-KBQA adopts a nonparametric approach and allows inspection of the retrieved nearest neighbors for a query. Moreover, one could *inject* a new relevant case into the case-memory (KNN-index), which could be picked up by the retriever and used by the reuse module to fix an erroneous prediction.

3.4.1 Performance on Unseen Relations

We consider the case when the model generates a wrong LF for a given query. We create a controlled setup by removing all queries from the training set of WebQSP which contain the (people.person.education) relation. This led to a removal of 136 queries from the train set and ensured that the model failed to correctly answer the 86 queries (held-out) in the test set which contained the removed relation in its LF.

We compare to a baseline transformer model (which do not use cases) as our baseline. As shown in Table 6, both baseline and CBR-KBQA do not perform well without any relevant cases since a required KB relation was missing during training. Next, we add the 136 training instances back to the training set and recompute the KNN index. This process involves encoding the newly added NL queries and recomputing the KNN index, a process which is computationally much cheaper than re-training the model again. Row 5 in Table 6


What colors do the school where Donald Stanley Marshall is grad student use?
Initial Prediction
Expert
Expert
Expert adds relevant simple case to the KNN index
What are Worcester Warrior's colors?
Model prediction after adding cases

```
SELECT DISTINCT ?x WHERE
?c educational_institution.students_graduates ?k
?c education.student Donald Stanley Marshall
?c educational_institution.mascot ?x
```

```
SELECT DISTINCT ?x WHERE
Worcester_Warrior
educational_institution.colors ?x
```

```
SELECT DISTINCT ?x WHERE
?c educational_institution.students_graduates ?k
?c education.student Donald Stanley Marshall
?c educational_institution.colors ?x
```

Figure 3: An expert point-fixes a model prediction by adding a simple case to the KNN index. Initial prediction was incorrect as no query with the relation (educational_institution.colors) was present in the train set. CBR-KBQA retrieves the case from the KNN index and fixes the erroneous prediction without requiring any re-training.

shows the new result. On addition of the new cases, CBR-KBQA can seamlessly use them and copy the unseen relation to predict the correct LF, reaching 70.6% accuracy on the 86 held-out queries.

In contrast, the baseline transformer must be fine-tuned on the new cases to handle the new relation, which is computationally more expensive than adding the cases to our index. Moreover, just fine-tuning on the new instances leads to *catastrophic forgetting* as seen in row 2 of Table 6 where the baseline model's performance on the initial set decreases drastically. We find it necessary to carefully fine-tune the model on new examples alongside original training examples (in a 1:2 proportion). However, it still converges to a performance which is lower than its original performance and much lower than the performance of CBR-KBQA.

3.4.2 Human-in-the-Loop Experiment

During error analysis, we realized that there are queries in the test set of WebQSP that contain KB relations in their LFs which were never seen during training[5]. That means models will never be able to predict the correct LF for the query because of the unseen relation. We conduct a human-in-the-loop experiment (Figure 3) in which users add *simple* 'cases' to point-fix erroneous predictions of CBR-KBQA for those queries. A simple case is a

[5]There are 94 different unseen relations in test set.

Scenario	Initial Set	Held-Out
Transformer	59.6	0.0
+ Fine-tune on additional cases only (100 gradient steps)	1.3	76.3
+ Fine-tune on additional cases and original data (300 gradient steps)	53.1	57.6
CBR-KBQA (Ours)	69.4	0.0
+ Adding additional cases to index (0 gradient steps; 2 sec)	69.4	70.6

Table 6: Robustness and controllability of our method against black-box transformers. CBR-KBQA can easily and quickly adopt to new relations given cases about it, whereas heavily parameterized transformer is not stable, and can undergo catastrophic forgetting when we try to add new relation information intro its parameters.

Scenario	P	R	F1	Acc
CBR-KBQA (Ours)	0.0	0.0	0.0	0.0
+ additional cases	36.54	38.59	36.39	32.89

Table 7: Results for H-I-T-L experiment. After adding a few cases, we see that we can get the accuracy of OOV questions to improve considerably, without needing to re-train the model.

Data	# Total Q	# Q that need comp. reasoning	# Correct	
			T5	CBR
CWQ	3531	639	205	**270**
CFQ	11968	6541	3351	**3886**

Table 8: We compare the performance of models on questions that need *novel combinations* of relations *not seen* during training.

NL query paired with a program which only contain one KB relation. Table 15 (Appendix D) shows some example of such cases. Because of the simple nature of the questions, these cases can be created manually (by a user who is knowledgeable about the KB schema) or automatically curated from data sources such as SimpleQuestions (Bordes et al., 2015) which is a large collection of NL queries that can be a mapped to a single KB edge. Table 14 in Appendix D shows various statistics of the missing relations and the number of cases added by humans and from SimpleQuestions. The cases are added to the original KNN-index. By adding a few cases, the performance increases from 0 to 36 F1 (Table 7) without requiring any training. Note unlike the previous controlled experiment in §3.4.1, we add around 3.87 cases for each unseen relation[6].

Importance of this result: We believe that flexibility of models to *fix* predictions (without training) is an important desideratum for QA systems deployed in production settings and we hope our results will inspire future research in this direction.

3.5 Further Analysis

We analyze questions in the evaluation set which require *novel combinations* of relations *never seen* in the training set. This means, in order for our model to answer these questions correctly, it would have to retrieve relevant nearest neighbor (NN) questions from the training set and copy the required relations from the logical form of *multiple* NN queries. Table 8 shows that our model outperforms the competitive T5 baseline. Also as we saw in the last section, our model is able to quickly adapt to relations *never seen* in the training set altogether by picking them up from newly added cases.

We also compare with a model with the same reuse component of CBR-KBQA but is trained and tested without retrieving any cases from the case-memory (Table 9). Even though the baseline model is competitive, having similar cases is beneficial, especially for the WebQSP dataset. We also report the results when we only use cross-entropy loss for training the BIGBIRD model and not the KL-divergence term. Table 10 reports the performance of CBR-KBQA using different number of retrieved cases. It is encouraging to see the the performance improves with increasing number of cases.

	WebQSP	CWQ
Baseline (K = 0)	67.2	65.8
CBR-KBQA (K = 20)	**69.9**	**67.1**
- KL term in loss	68.1	66.7

Table 9: Ablation experiment with a baseline model that do not use cases and also when the KL divergence term (§2.2) is not used in loss function of reuse step . The numbers denote exact match accuracy.

# nearest neighbors	Accuracy
K = 0	67.20
K = 1	68.45
K = 10	69.23
K = 20	69.98

Table 10: Performance on WebQSP on varying number of nearest neigbors

4 Related Work

Retrieval augmented QA models (Chen et al., 2017; Guu et al., 2020; Lewis et al., 2020b) augments a reader model with a retriever to find rel-

[6]In §3.4.1, we added 136 cases (v/s 3.87) for one relation. This is why the accuracy in Table 6 is higher w.r.t Table 7.

evant paragraphs from a nonparametric memory. In contrast, our CBR approach retrieves similar queries and uses their logical forms to derive a new solution. Recently Lewis et al. (2020c) proposed a model that finds a nearest neighbor (NN) question in the training set and returns the answer to that question. While this model would be helpful if the exact question or its paraphrase is present in the training set, it will not generalize to other scenarios. CBR-KBQA, on the other hand, learns to reason with the retrieved programs of multiple retrieved NN queries and generates a new program for the given query and hence is able to generalize even if the query paraphrase is not present in the train set.

Retrieve and edit: CBR-KBQA shares similarities with the RETRIEVE-AND-EDIT framework (Hashimoto et al., 2018) which utilizes retrieved nearest neighbor for structured prediction. However, unlike our method they only retrieve a single nearest neighbor and will unlikely be able to generate programs for questions requiring relations from multiple nearest neighbors.

Generalizing to unseen database schemas: There has been work in program synthesis that generates SQL programs for unseen database schemas (Wang et al., 2020; Lin et al., 2020). However, these work operate on web or Wikipedia tables with small schemas. For example, in WikiTableQuestions (Pasupat and Liang, 2015) the average. number of columns in a table is 5.8 and in Spider dataset (Yu et al., 2018), the average number of columns is 28.1. On the other hand, our model has to consider all possible Freebase relations (in thousands). Previous work perform schema-aware encoding which is not possible in our case because of the large number of relations. The retrieve step of CBR-KBQA can be seen as a pruning step which narrows the number of candidate relations by retrieving relevant questions and their logical forms.

Case-based Reasoning for KB completion: Recently, a CBR based KB reasoning approach was proposed by Das et al. (2020a,b). They retrieve similar entities and then find KB reasoning paths from them. However, their approach does not handle complex natural language queries and only operate on structured triple queries. Additionally, the logical forms handled by our model have much more expressive power than knowledge base paths.

Program Synthesis and Repair: Repairing / revising generated programs has been studied in the field of program synthesis. For example, prior work repairs a program based on syntax of the underlying language (Le et al., 2017), by generating sketches (Hua et al., 2018). More recently, Gupta et al. (2020) proposes a framework in which they use a program debugger to revise the program generated by a neural program synthesizer. However, none of these works take advantage of the similarity between semantic relations present in the knowledge base, and hence, unlike us, they do not use embeddings of similar relation to align relations. More generally, many prior efforts have employed neural models to generate SPARQL-like code for semantic parsing (Dong and Lapata, 2016; Balog et al., 2016; Zhong et al., 2017a), SQL queries over relational databases (Zhong et al., 2017b), program-structured neural network layouts (Andreas et al., 2016), or even proofs for mathematical theorems (Polu and Sutskever, 2020). Our work differs in our use of the programs of multiple retrieved similar queries to generate the target program.

K-NN approach in other NLP applications: Khandelwal et al. (2020) demonstrate improvements in language modeling by utilizing explicit examples from training data. There has been work in machine translation (Zhang et al., 2018; Gu et al., 2018; Khandelwal et al., 2021) that uses nearest neighbor translation pair to guide the decoding process. Recently, Hossain et al. (2020) proposed a retrieve-edit-rerank approach for text generation in which each retrieved candidate from the training set is edited independently and then re-ranked. In contrast, CBR-KBQA generates the program *jointly* from all the retrieved cases and is more suitable for questions which needs copying relations from multiple nearest neighbors. Please refer to (§E) for further related work.

5 Limitations and Future Work

To the best of our knowledge, we are the first to propose a neuralized CBR approach for KBQA. We showed that our model is effective in handling complex questions over KBs, but our work also has several limitations. First, our model relies on the availability of supervised logical forms such as SPARQL queries, which can be expensive to annotate at scale. In the future, we plan to explore ways to directly learn from question-answer pairs (Berant et al., 2013; Liang et al., 2016). Even though, CBR-KBQA is modular and has several advantages, the retrieve and reuse components of our model are trained separately. In future, we plan to explore avenues for end to end learning for CBR.

Acknowledgments

We thank Prof. Yu Su and Yu Gu (Ohio State University) for their help in setting up the Freebase server, Andrew Drozdov, Kalpesh Krishna, Subendhu Rongali and other members of the UMass IESL and NLP groups for helpful discussion and feedback. RD and DT are funded in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction and in part by IBM Congitive Horizons Network (CHN). EP is grateful for support from the NSF Graduate Research Fellowship and the Open Philanthropy AI Fellowship. The work reported here was performed in part by the Center for Data Science and the Center for Intelligent Information Retrieval, and in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative.

References

Agnar Aamodt and Enric Plaza. 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Matej Balog, Alexander L Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. 2016. Deepcoder: Learning to write programs. *arXiv preprint arXiv:1611.01989*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *ICDM*.

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

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Neurips*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *ACL*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven Gillis. 1996. Mbt: A memory-based part of speech tagger-generator. In *WVLC*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. In *ICLR*.

Rajarshi Das, Ameya Godbole, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2020a. A simple approach to case-based reasoning in knowledge bases. In *AKBC*.

Rajarshi Das, Ameya Godbole, Nicholas Monath, Manzil Zaheer, and Andrew McCallum. 2020b. Probabilistic case-based reasoning for open-world knowledge graph completion. In *Findings of EMNLP*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *ACL*.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *arXiv preprint arXiv:2007.08970*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *AAAI*.

Kavi Gupta, Peter Ebert Christensen, Xinyun Chen, and Dawn Song. 2020. Synthesize, execute and debug: Learning to repair for neural program synthesis. In *Neurips*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. In *ICML*.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Neurips*.

Geoffrey E Hinton and David C Plaut. 1987. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*.

Nabil Hossain, Marjan Ghazvininejad, and Luke Zettlemoyer. 2020. Simple and effective retrieve-edit-rerank text generation. In *ACL*.

Jinru Hua, Mengshi Zhang, Kaiyuan Wang, and Sarfraz Khurshid. 2018. Towards practical program repair with on-demand candidate generation. In *international conference on software engineering*, pages 12–23.

Yuncheng Hua, Yuan-Fang Li, Reza Haffari, Guilin Qi, and Wei Wu. 2020. Retrieve, program, repeat: complex knowledge base question answering via alternate meta-learning. In *International Joint Conference on Artificial Intelligence 2020*, pages 3679–3686. Association for the Advancement of Artificial Intelligence (AAAI).

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *ICLR*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *ICLR*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *ICLR*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *PNAS*.

Janet L Kolodner. 1983. Maintaining organization in a dynamic long-term memory. *Cognitive science*.

Brenden M Lake and Marco Baroni. 2018. Generalization without systematicity. In *ICML*.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *ACL*.

Yunshi Lan, Shuohang Wang, and Jing Jiang. 2019. Knowledge base question answering with topic units.

Juliana S Lancaster and Janet L Kolodner. 1987. Problem solving in a natural task as a function of experience. Technical report, Georgia Tech CS Department.

Xuan-Bach D Le, Duc-Hiep Chu, David Lo, Claire Le Goues, and Willem Visser. 2017. S3: syntax-and semantic-guided repair synthesis via programming by examples. In *Foundations of Software Engineering*.

David B Leake. 1996. Cbr in context: The present and future. *Case-based reasoning: Experiences, lessons, and future directions*.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Neurips*.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2020c. Question and answer test-train overlap in open-domain question answering datasets. *arXiv preprint arXiv:2008.02637*.

Belinda Z Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *EMNLP*.

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*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. In *Findings of EMNLP*.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D Ernst. 2018. Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system. In *LREC*.

Joao Loula, Marco Baroni, and Brenden M Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *EMNLP Blackbox NLP Workshop*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL-HLT*.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, Florence, Italy. Association for Computational Linguistics.

Ngonga Ngomo. 2018. 9th challenge on question answering over linked data (qald-9). *language*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Gaurav Pandey, Danish Contractor, Vineet Kumar, and Sachindra Joshi. 2018. Exemplar encoder-decoder for neural conversation generation. In *ACL*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *ACL*.

Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. In *NAACL*.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8864–8880, Online. Association for Computational Linguistics.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Edwina L Rissland. 1983. Examples in legal reasoning: Legal hypotheticals. In *IJCAI*.

Brian H Ross. 1984. Remindings and their effects in learning a cognitive skill. *Cognitive psychology*.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *ACL*.

Roger C Schank. 1982. *Dynamic memory: A theory of reminding and learning in computers and people*. cambridge university press.

Henk Schmidt, Geoffrey Norman, and Henny Boshuizen. 1990. A cognitive perspective on medical expertise: theory and implications. *Academic medicine*.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2020. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? *arXiv preprint arXiv:2010.12725*.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *ACL*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Neurips*.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *EMNLP*.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. In *ACL*.

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

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. In *EMNLP*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019b. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *NAACL-HLT*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Continuous Vector Space Models and their Compositionality Workshop*.

Petar Velickovic, Guillem Cucurull, A. Casanova, A. Romero, P. Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *ACL*.

Jason Weston, Emily Dinan, and Alexander Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. In *ConvAI Workshop EMNLP*.

Sam Wiseman and Karl Stratos. 2019. Label-agnostic sequence labeling by copying nearest neighbors. In *ACL*.

Tomer Wolfson, Mor Geva, Ankit Gupta, Yoav Goldberg, Matt Gardner, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

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 *ACL*.

Dong Yu, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide. 2013. Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *ICASSP*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *Neurips*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *NCAI*.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding neural machine translation with retrieved translation pieces. In *NAACL*.

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

Victor Zhong, Caiming Xiong, and Richard Socher. 2017b. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Dataset	Train	Valid	Test
WebQSP	2,798	300	1,639
CWQ	27,639	3,519	3,531
CFQ	95,743	11,968	11,968

Table 11: Dataset statistics

Dataset	Validation Acc
WebQSP	71.5
CWQ	82.8
CFQ	69.9

Table 12: Validation set accuracy of models corresponding to the results reported in the paper

A EMNLP Reproducibility Checklist

A.1 Data

WebQSP contains 4737 NL questions belonging to 56 domains covering 661 unique relations. Most questions need up to 2 hops of reasoning, where each hop is a KB edge. COMPLEXWEBQUESTIONS (CWQ) is generated by extending the WebQSP dataset with the goal of making it a more complex multi-hop dataset. There are four types of questions: composition (45%), conjunction (45%), comparative (5%), and superlative (5%). Answering these questions requires up to 4 hops of reasoning in the KB, making the dataset challenging. Compositional Freebase Questions (CFQ) is a recently proposed benchmark explicitly developed for measuring compositional generalization. For all the datasets above, the logical form (LF) for each NL question is a SPARQL query that can be executed against the Freebase KB to obtain the answer entity.

A.2 Hyperparameters

The WebQSP dataset does not contain a validation split, so we choose 300 training instances to form the validation set. We use grid-search (unless explicitly mentioned) to set the hyperparameters listed below.

Case Retriever: We initialize our retriever with the pre-trained ROBERTA-base weights. We set the initial learning rate to 5×10^{-5} and decay it linearly throughout training. We evaluate the retriever based on the percentage of gold LF relations in the LFs of the top-k retrieved cases (recall@k). We train for 10 epochs and use the best checkpoint based on recall@20 on the validation set. We set train and validation batch sizes to 32.

For p_{mask}, we try values from [0, 0.2, 0.4, 0.5, 0.7, 0.9, 1]. When training the retriever, we found $p_{\text{mask}} = 0.2$ works best for COMPLEXWEBQUESTIONS and $p_{\text{mask}} = 0.5$ for the remaining datasets.

Seq2Seq Generator: We use a BIGBIRD generator network with 6 encoding and 6 decoding sparse-attention layers, which we initialize with pre-trained BART-base weights. We set the initial learning rate to 5×10^{-5} and decay it linearly throughout training. Accuracy after the execution of generated programs on the validation set is used to select the optimal setting and model checkpoint.

For λ_T, we perform random search in range [0, 1]. We finally use λ_T=1.0 for all datasets. For k (number of cases), we search over the values [1, 3, 5, 7, 10, 20]. For all datasets, we use k=20 cases and decode with a beam size of 5 for decoding. The WebQSP model was trained for 15K gradient steps and all other models were trained for 40K gradient steps.

Computing infrastructure: We perform our experiments on a GPU cluster managed by SLURM. The case retriever was trained and evaluated on NVIDIA GeForce RTX 2080 Ti GPU. The models for the Reuse step were trained and evaluated on NVIDIA GeForce RTX 8000 GPUs. Revise runs on NVIDIA GeForce RTX 2080 Ti GPU when using ROBERTA for alignment and runs only on CPU when using TRANSE. We report validation set scores in Table 12.

B Further Experiments and Analysis

B.1 Performance of Retriever

We compare the performance of our trained retriever with a ROBERTA-base model. We found that ROBERTA model even without any fine-tuning performs well at retrieval. However, fine-tuning ROBERTA with our distant supervision objective improved the overall recall, e.g., from 86.6% to 90.4% on WEBQUESTIONSSP and from 94.8% to 98.4% on CFQ.

B.2 Performance on Unseen Entities

In Table 7 we showed CBR-KBQA is effective for unseen relations. But what about unseen entities in the test set?. On analysis we found that in WebQSP, CBR-KBQA can copy unseen entities correctly **86.8%** (539/621) from the question. This is +1.9% improvement from baseline transformer model which is able to copy correctly 84.9% (527/621) of the time. Note that unseen entities can be copied from the input NL query and we do not need additional cases to be injected to KNN index.

B.3 Analysis of the Revise Step

In the revise step, we attempt to fix programs predicted by our reuse step that did not execute on the knowledge base. The predicted program can be syntactically incorrect or enforce conditions that lead to an unsatisfiable query. In our work, we focus on predicted programs that can be fixed by aligning clauses to relations in the local neighborhood of query entities. We give examples of successful alignments Table 16 as well as failed attempts at alignment Table 17.

C Details on Held-Out Experiments

In this section, we include more details about our held-out experiment described in section 3.4.1. The goal of this experiment is to show that our approach can *generalize* to unseen relations without requiring *any further training* of the model. This is a relevant setting to explore, because real-world knowledge bases are often updated with new kinds of relations, and we would like KBQA systems that adapt to handle new information with minimal effort.

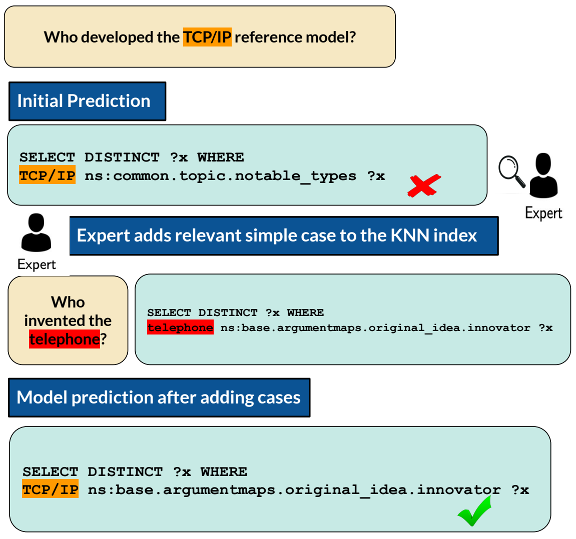
We explicitly hold-out all questions containing a particular relation from the datasets. Table 13 shows the relation type and the number of questions that are removed as a result of removing the relation.

Dataset	Relation name	Train	Test
WebQSP	people.person.education	136	86

Table 13: Relation type and the corresponding number of NL queries that are held-out.

D Details on Automated Case Collection and Human-in-the-Loop Experiments

While conducting analysis, we also noticed that WebQSP has queries in the test set for which the required relations are never present in the training set. This gives us an opportunity to conduct real human-in-the-loop experiments to demonstrate the advantage of our model. To add more cases, we resort to a mix of automated data collection and human-in-the-loop strategy. For each of the missing relation, we first try to find NL queries present


Who developed the TCP/IP reference model?
Initial Prediction
Expert
Expert adds relevant simple case to the KNN index
Expert
Who invented the telephone?
Model prediction after adding cases

```
SELECT DISTINCT ?x WHERE
TCP/IP ns:common.topic.notable_types ?x
```

```
SELECT DISTINCT ?x WHERE
telephone ns:base.argumentmaps.original_idea.innovator ?x
```

```
SELECT DISTINCT ?x WHERE
TCP/IP ns:base.argumentmaps.original_idea.innovator ?x
```

Figure 4: An example query where our approach correctly utilizes added H-I-L-T cases

in the SimpleQuestions (Bordes et al., 2015). SimpleQuestions (SQ) is a large dataset containing more than 100K NL questions that are 'simple' in nature — i.e. each NL query maps to a single relation (fact) in the Freebase KB. For each missing relation type, we try to find questions in the SQ dataset that can be mapped to the missing relation. However, even SQ has missing coverage in which case, we manually generate a question and its corresponding SPARQL query by reading the description of the relation. Table 14 shows the number of questions in the evaluation set which at least has a relation never seen during training and also the number of cases that has been added. For example, we[7] were able to collect 292 questions from SQ and we manually created 72 questions for WebQSP. Overall, we add 3.87 new cases per query relation for WebQSP.

Table 15 shows some example of cases added manually or from SQ. We look up entity ids for entities from the FACC1 alias table (§3.1). Also note, that since we only add questions which are simple in nature, the corresponding SPARQL query can be easily constructed from the missing relation type and the entity id.

Importance of this result: Through this experiment, we demonstrate two important properties of our model — *interpretability* and *controllability*. Database schemas keep changing and new tables keep getting added to a corporate database. When our QA system gets a query wrong, by looking at the retrieved K-nearest neighbors, users can deter-

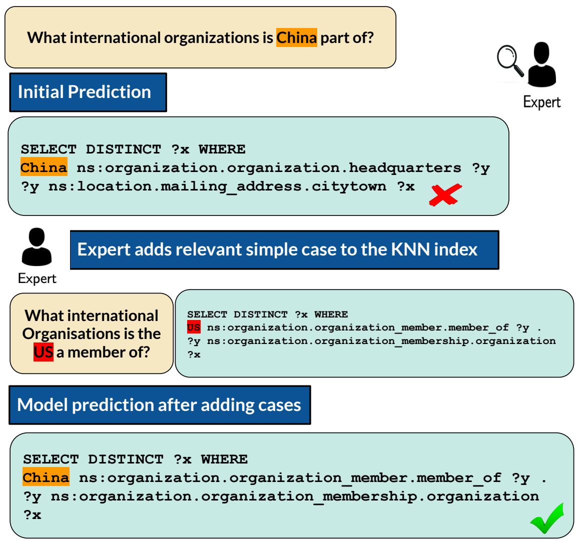
[7]The H-I-T-L case addition was done by 2 graduate students in the lab.

Dataset	# missing relations	# questions	Cases Added via		Avg. # cases per relation
			H-I-T-L	SimpleQuestions	
WebQSP	94	79	72	292	3.87

Table 14: Number of questions in the evaluation set that needs a relation which is not seen in the training set. Note that, there can be multiple relations in a question that might not be seen during training. The last two columns show the number of cases added both via human-in-the-loop (H-I-T-L) annotation and automatically from SimpleQuestions dataset.

NL Query	SPARQL	Source
What is the Mexican Peso called?	select ?x where { m.012ts8 finance.currency.currency_code ?x .}	Manual
Who invented the telephone?	select ?x where { m.0g_3r base.argumentmaps.original_idea.innovator ?x .}	Manual
what area is wrvr broadcated in?	select ?x where { m.025z9rx broadcast.broadcast.area_served ?x .}	SQ
Where are Siamese cats originally from?	select ?x where { m.012ts8 biology.animal_breed.place_of_origin ?x .}	Manual

Table 15: Examples of few added questions and their corresponding SPARQL queries. Notice that the SPARQL queries are very simple to create once we know the name of the missing relation. The source column indicate whether the question was manually created or automatically added from Simple Questions (SQ) dataset.



What international organizations is China part of?
Expert
Initial Prediction
Expert
Expert adds relevant simple case to the KNN index
What international Organisations is the US a member of?
Model prediction after adding cases

```
SELECT DISTINCT ?x WHERE
China ns:organization.organization.headquarters ?y
?y ns:location.mailing_address.citytown ?x
```

```
SELECT DISTINCT ?x WHERE
US ns:organization.organization_member.member_of ?y .
?y ns:organization.organization_membership.organization
?x
```

```
SELECT DISTINCT ?x WHERE
China ns:organization.organization_member.member_of ?y .
?y ns:organization.organization_membership.organization
?x
```

Figure 5: An example query where our approach correctly utilizes added H-I-L-T cases

mine (interpretability) that the required relation is not present in the training set. By adding few cases for the new relations, they can query the DB for similar questions, without needing to train the QA system (controllability). Current black-box NLP models are not capable of doing such point-fixes and our experiment is an initial attempt towards building such systems.

E Further Related Work

KNN approach in other NLP applications (continued): Wiseman and Stratos (2019) achieved accurate sequence labeling by explicitly and only copying labels from retrieved neighbors. NN models have been used in a numbe NLP applications such as POS tagging (Daelemans et al., 1996). Another recent line of work use training examples at test time to improve language generation (Weston et al., 2018; Pandey et al., 2018; Cao et al., 2018; Peng et al., 2019). Hua et al. (2020) recently proposed a meta-learning approach which utilizes cases retrieved w.r.t. the similarity of the input. However, their main goal is to learn a better parametric model (retriever and generator) from neighboring cases rather than composing and fixing cases to generate answers at test time.

Question Decomposition One strategy to answer a complex question is to first break it down into simpler subquestions, each of which can be viewed as a natural language program describing how to answer the question. This approach has been shown to be effective as far back as IBM Watson (Ferrucci et al., 2010) to more recent systems for answering questions about text (Das et al., 2019; Min et al., 2019; Perez et al., 2020; Wolfson et al., 2020) or knowledge bases (Talmor and Berant, 2018). These prior studies do not leverage case-based reasoning when generating decompositions and thus may also benefit from similar techniques as proposed in our work.

	WebQSP
Question:	when did kaley cuoco m.03kxp7 join charmed m.01f3p_ ?

Predicted SPARQL:	SELECT DISTINCT ?x WHERE {
	ns:m.03kxp7 ns:tv.tv_character.appeared_in_tv_program ?y .
	?y ns:tv.regular_tv_appearance.from ?x .
	?y ns:tv.regular_tv_appearance.series ns:m.01f3p_ .
	}

Ground-truth SPARQL:	SELECT DISTINCT ?x WHERE {
	ns:m.03kxp7 ns:tv.tv_actor.starring_roles ?y .
	?y ns:tv.regular_tv_appearance.from ?x .
	?y ns:tv.regular_tv_appearance.s eries ns:m.01f3p_ .
	}

Revised SPARQL:	SELECT DISTINCT ?x WHERE {
	ns:m.03kxp7 **ns:tv.tv_actor.starring_roles** ?y .
	?y ns:tv.regular_tv_appearance.from ?x .
	?y ns:tv.regular_tv_appearance.s eries ns:m.01f3p_ .
	}

	CWQ
Question:	What text in the religion which include Zhang Jue m.02gjv7 as a key figure is considered to be sacred m.02vt2rp ?

Predicted SPARQL:	SELECT DISTINCT ?x WHERE {
	?c ns:religion.religion.deities ns:m.02gjv7 .
	?c ns:religion.religion.texts ?x .
	. . . *benign filters* . . . }

Ground-truth SPARQL:	SELECT DISTINCT ?x WHERE {
	?c ns:religion.religion.notable_figures ns:m.02gjv7 .
	?c ns:religion.religion.texts ?x .}

Revised SPARQL:	SELECT DISTINCT ?x WHERE {
	?c **ns:religion.religion.notable_figures** ns:m.02gjv7 .
	?c ns:religion.religion.texts ?x .
	. . . *benign filters* . . . }

Question:	What is the mascot of the educational institution that has a sports team named the North Dakota State Bison m.0c5s26 ?

Predicted SPARQL:	SELECT DISTINCT ?x WHERE {
	?c ns:education.educational_institution.sports_teams ns:m.0c5s26 .
	?c ns:education.educational_institution.mascot ?x .
	}

Ground-truth SPARQL:	SELECT DISTINCT ?x WHERE {
	?c ns:education.educational_institution.sports_teams ns:m.0c41_v .
	?c ns:education.educational_institution.mascot ?x .
	}

Revised SPARQL:	SELECT DISTINCT ?x WHERE {
	?c **ns:education.educational_institution.athletics_brand** ns:m.0c5s26 .
	?c ns:education.educational_institution.mascot ?x .
	}

Comments:	The entity linker has tagged the bison as a university symbol (m.0c5s26) rather than the Bison football team (m.0c41_v). Alignment helps the model recover from this by picking the relation that connects the tagged entity to the university.

Table 16: Examples of successful alignment with TransE from the Revise stage.

	WebQSP
Question:	who is gimli m.0h34n 's father m.02pn7 in the hobbit m.0n4ck66
Predicted SPARQL:	SELECT DISTINCT ?x WHERE { ns:m.0h34n ns:people.person.parents ?x . ?x ns:people.person.gender ns:m.02pn7 . ?x ns:people.person.parents ?sk0 . } ORDER BY xsd:datetime(?sk0) LIMIT 1

Ground-truth SPARQL:	SELECT DISTINCT ?x WHERE {
	ns:m.0h34n ns:fictional_universe.fictional_character.parents ?x .
	?x ns:fictional_universe.fictional_character.gender ns:m.05zppz .
	}

Predicted SPARQL:	SELECT DISTINCT ?x WHERE { ns:m.0h34n **ns:fictional_universe.fictional_character.parents** ?x . ?x ns:people.person.gender ns:m.02pn7 . ?x ns:people.person.parents ?sk0 . } ORDER BY xsd:datetime(?sk0) LIMIT 1
Comments:	In this example the prediction has an incorrect structure, so aligning an edge does not change the outcome.
	CWQ
Question:	What political leader runs the country where the Panama m.05qx1 nian Balboa m.0200cp is used?
Predicted SPARQL:	SELECT DISTINCT ?x WHERE { ?c ns:location.country.currency_formerly_used ns:m.0200cp . ?c ns:government.governmental_jurisdiction.governing_officials ?y . ?y ns:government.government_position_held.office_holder ?x . . . . *benign filters* . . . }
Ground-truth SPARQL:	SELECT DISTINCT ?x WHERE { ?c ns:location.country.currency_used ns:m.0200cp . ?c ns:government.governmental_jurisdiction.governing_officials ?y . ?y ns:government.government_position_held.office_holder ?x . **?y ns:government.government_position_held.office_position_or_title ns:m.0m57hp6** . . . . *benign filters* . . . }
Revised SPARQL:	SELECT DISTINCT ?x WHERE { ?c **ns:location.country.currency_used** ns:m.0200cp . ?c ns:government.governmental_jurisdiction.governing_officials ?y . ?y ns:government.government_position_held.office_holder ?x . . . . *benign filters* . . . }
Target Answers:	{m.06zmv9x}
Revised Answers:	{m.02y8_r, m.06zmv9}
Comments:	The original prediction has missing clauses so alignment produces more answers than target program

Table 17: Examples of failed alignment with TransE from the Revise stage.