What's Missing: A Knowledge Gap Guided Approach for Multi-hop Question Answering

Tushar Khot and **Ashish Sabharwal** and **Peter Clark**
Allen Institute for Artificial Intelligence, Seattle, WA, U.S.A.
{tushark,ashishs,peterc}@allenai.org

Abstract

Multi-hop textual question answering requires combining information from multiple sentences. We focus on a natural setting where, unlike typical reading comprehension, only *partial information* is provided with each question. The model must retrieve and use additional knowledge to correctly answer the question. To tackle this challenge, we develop a novel approach that explicitly identifies the *knowledge gap* between a key span in the provided knowledge and the answer choices. The model, GapQA, learns to fill this gap by determining the relationship between the span and an answer choice, based on retrieved knowledge targeting this gap. We propose jointly training a model to simultaneously fill this knowledge gap and compose it with the provided partial knowledge. On the OpenBookQA dataset, given partial knowledge, explicitly identifying what's missing substantially outperforms previous approaches.

1 Introduction

Reading Comprehension datasets (Richardson et al., 2013; Rajpurkar et al., 2016; Joshi et al., 2017) have gained interest as benchmarks to evaluate a system's ability to understand a document via question answering (QA). Since many of these early datasets only required a system to understand a single sentence, new datasets were specifically designed to focus on the problem of multi-hop QA, i.e., reasoning across sentences (Khashabi et al., 2018; Welbl et al., 2018; Yang et al., 2018).

While this led to improved language understanding, the tasks still assume that a system is provided with *all* knowledge necessary to answer the question. In practice, however, we often only have access to *partial knowledge* when dealing with such multi-hop questions, and must retrieve additional facts (the knowledge "gaps") based on the question and the provided knowledge. Our goal is to identify such gaps and fill them using an external knowledge source.

Question:
Which of these would let the most heat travel through?
A) a new pair of jeans.
B) a steel spoon in a cafeteria.
C) a cotton candy at a store.
D) a calvin klein cotton hat.
Core Fact:
Metal lets heat travel through.

Knowledge Gap (similar gaps for other choices):
steel spoon in a cafeteria _____ metal.

Filled Gap (relation identified using KB):
steel spoon in a cafeteria *is made of* metal.

Figure 1: A sample OpenBookQA question, the identified knowledge gap based on partial information in the core fact, and relation (*is made of*) identified from a KB to fill that gap.

The recently introduced challenge of *open book* question answering (Mihaylov et al., 2018) highlights this phenomenon. The questions in the corresponding dataset, OpenBookQA, are derived from a science fact in an "open book" of about 1300 facts. To answer these questions, a system must not only identify a relevant "core" science fact from this small book, but then also retrieve additional common knowledge from large external sources in order to successfully apply this core fact to the question. Consider the example in Figure 1. The core science fact *metal lets heat to travel through* points to *metal* as the correct answer, but it is not one of the 4 answer choices. Given this core fact (the "partial knowledge"), a system must still use broad external knowledge to fill the remaining gap, that is, identify which answer choice *contains* or *is made of* metal.

This work focuses on *QA under partial knowledge*. This turns out to be a surprisingly chal-

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 2814–2828, Hong Kong, China, November 3–7, 2019.

lenging task in itself; indeed, the partial knowledge models of Mihaylov et al. (2018) achieve a score of only 55% on OpenBookQA, far from human performance of 91%. Since this and several recent multi-hop datasets use the multiple-choice setting (Welbl et al., 2018; Khashabi et al., 2018; Lai et al., 2017), we assume access to potential answers to a question. While our current model relies on this for a direct application to span-prediction based RC datasets, the idea of identifying knowledge gaps can be used to create novel RC specific models.

We demonstrate that an intuitive approach leads to a strong model: first identify the knowledge gap and then fill this gap, i.e., identify the missing relation using external knowledge. We primarily focus on the OpenBookQA dataset since it is the only dataset currently available that provides partial context. However, we believe such an approach is also applicable to the broader setting of multi-hop RC datasets, where the system could start reasoning with one sentence and fill remaining gap(s) using sentences from other passages.

Our model operates in two steps. First, it predicts a key span in the core fact ("metal" in the above example). Second, it answers the question by identifying the relationship between the key span and answer choices, i.e., by *filling* the knowledge gap. This second step can be broken down further: (a) retrieve relevant knowledge from resources such as ConceptNet (Speer et al., 2017) and large-scale text corpora (Clark et al., 2018); (c) based on this, predict potential relations between the key span and an answer choice; and (d) compose the core fact with this filled gap.

We collect labels for knowledge gaps on ∼30% of the training questions, and train two modules capturing the two main steps above. The first exploits an existing RC model and large-scale dataset to train a span-prediction model. The second uses multi-task learning to train a separate QA model to jointly predict the relation representing the gap, as well as the final answer. For questions without labelled knowledge gaps, the QA model is trained based solely on the predicted answer.

Our model outperforms the previous state-of-the-art partial knowledge models by 6.5% (64.41 vs 57.93) on a targeted subset of OpenBookQA amenable to gap-based reasoning. Even without missing fact annotations, our model with a simple heuristic to identify missing gaps still outperforms previous models by 3.4% (61.38 vs. 57.93). It also generalizes to questions that were not its target, with 3.6% improvement (59.40 vs. 55.84) on the full OpenBookQA set.

Overall, the contributions of this work are: (1) an analysis and dataset[1] of knowledge gaps for QA under partial knowledge; (2) a novel two-step approach of first identifying and then filling knowledge gaps for multi-hop QA; (3) a model[1] that simultaneously learns to fill a knowledge gap using retrieved external knowledge and compose it with partial knowledge; and (4) new state-of-the-art results on QA with partial knowledge (+6.5% using annotations on only 30% of the questions).

2 Related Work

Text-Based QA. Reading Comprehension (RC) datasets probe language understanding via question answering. While several RC datasets (Rajpurkar et al., 2016; Trischler et al., 2017; Joshi et al., 2017) can be addressed with single sentence understanding, newer datasets (Weston et al., 2015; Welbl et al., 2018; Khashabi et al., 2018; Yang et al., 2018) specifically target multi-hop reasoning. In both cases, all relevant information, barring some linguistic knowledge, is provided or the questions are unanswerable (Rajpurkar et al., 2018). This allows using an attention-based approach of indirectly combining information (Dhingra et al., 2018; Cao et al., 2019; Song et al., 2018).

On the other hand, open domain question answering datasets (Clark et al., 2016, 2018) come with no context, and require first retrieving relevant knowledge before reasoning with it. Retrieving this knowledge from noisy textual corpora, while simultaneously solving the reasoning problem, can be challenging, especially when questions require multiple facts. This results in simple approaches (e.g. word-overlap/PMI-based approaches), that do not heavily rely on the retrieval quality, being competitive with other complex reasoning methods that assume clean knowledge (Clark et al., 2016; Jansen et al., 2017; Angeli et al., 2016). To mitigate this issue, semi-structured tables (Khashabi et al., 2016; Jansen et al., 2018) have been manually authored targeting a subset of these questions. However, these tables are expensive to create and these questions often need multiple hops (sometimes up to

[1]The code and associated dataset are available at https://github.com/allenai/missing-fact.

16 (Jansen et al., 2018)), making reasoning much more complex.

OpenBookQA dataset (Mihaylov et al., 2018) was proposed to limit the retrieval problem by providing a set of ∼1300 facts as an 'open book' for the system to use. Every question is based on one of the core facts, and in addition requires basic external knowledge such as hypernymy, definition, and causality. We focus on the task of question answering under partial context, where the core fact for each question is available to the system.

Knowledge-Based QA. Another line of research is answering questions (Bordes et al., 2015; Pasupat and Liang, 2015; Berant et al., 2013) over a structured knowledge base (KB) such as Freebase (Bollacker et al., 2008). Depending on the task, systems map questions to a KB query with varying complexity: from complex semantic parses (Krishnamurthy et al., 2017) to simple relational lookup (Petrochuk and Zettlemoyer, 2018). Our sub-task of filling the knowledge gap can be viewed as KB QA task with knowledge present in a KB or expected to be inferred from text.

Some RC systems (Mihaylov and Frank, 2018; Kadlec et al., 2016) and Textual Entailment (TE) models (Weissenborn et al., 2017; Inkpen et al., 2018) incorporate external KBs to provide additional context to the model for better language understanding. However, we take a different approach of using this background knowledge in an explicit inference step (i.e. hop) as part of a multi-hop QA model.

3 Knowledge Gaps

We now take a deeper look at categorizing knowledge gaps into various classes. While grounded in OpenBookQA, this categorization is relevant for other multi-hop question sets as well.[2] We will then discuss how to effectively annotate such gaps.

3.1 Understanding Gaps: Categorization

We analyzed the additional facts needed for answering 75 OpenBookQA questions. These facts naturally fall into three classes, based on the knowledge gap they are trying to fill: (1) Question-to-Fact, (2) Fact-to-Answer, and (3) Question-to-Answer(Fact). Figure 2 shows a high-level overview, with simplified examples of each class of knowledge gap in Figures 3, 4, and 5.

[2] As mentioned earlier, in the RC setting, the first relevant sentence read by the system can be viewed as the core fact.

Question-to-Fact Gap. This gap exists between concepts in the question and the core fact. For example, in Figure 3, the knowledge that "Kool-aid" is a liquid is needed to even recognize that the fact is relevant.

Fact-to-Answer Gap. This gap captures the relationship between concepts in the core fact and the answer choices. For example, in Figure 4, the knowledge "Heat causes evaporation" is needed to relate "evaporated" in the fact to the correct answer "heat". Note that it is often possible to find relations connecting the fact to even incorrect answer choices. For example, "rainfall" could be connected to the fact using "evaporation leads to rainfall". Thus, identifying the correct relation and knowing if it can be composed with the core fact is critical, i.e., "evaporation causes liquid to disappear" and "evaporation leads to rainfall" do not imply that "rainfall causes liquid to disappear".

Question-to-Answer(Fact) Gap. Finally, some questions need additional knowledge to connect concepts in the question to the answer, based on the core fact. For example, composition questions (Figure 5) use the provided fact to replace parts of the original question with words from the fact.

Notably Question-to-Fact and Fact-to-Answer gaps are more common in OpenBookQA (44% and 86% respectively[3]), while the Question-to-Answer(Fact) gap is very rare (<20%). While all three gap classes pose important problems, we focus on Fact-to-Answer gap and assume that the core fact is provided. This is still a challenging problem as one must not only identify and fill the gap, but also learn to compose this filled gap with the input fact.

3.2 Annotating Gaps: Data Collection

Due to space constraints, details of our crowdsourcing process for annotating knowledge gaps, including the motivation behind various design choices as well as several examples, are deferred to the Appendix (Section A). Here we briefly summarize the final crowdsourcing design.

Our early pilots revealed that straightforward approaches to annotate knowledge gaps for all OpenBookQA questions lead to noisy labels. To address this, we (a) identified a **subset of questions** suitable for this annotation task and (b) split

[3] Some questions have both of these classes of gaps.

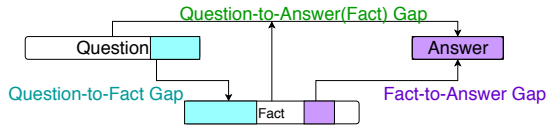

Figure 2: High-level overview of the kinds of knowledge gaps, assuming partial knowledge from the fact. In subsequent figures, the knowledge gap is indicated using highlighted text.

Question: What can cause liquid to disappear ?
Fact: If a liquid disappears then that liquid probably evaporated
Answer: Heat
Gap: Heat causes evaporation

Figure 4: Knowledge gap between the **fact** (evaporated) and the **answer** (Heat). While it is clear how to apply the knowledge, we need to know that "Heat causes evaporation" to identify the right answer.

Question: What can cause Kool-aid to disappear ?
Fact: If a liquid disappears then that liquid probably evaporated
Answer: Evaporation
Gap: Kool-aid is a liquid

Figure 3: Knowledge gap between the **question** (Kool-aid) and the **fact** (liquid). To apply the fact about liquids to the question, we need to know "Kool-aid is a liquid".

Question: What is the satellite of the blue planet ?
Fact: The blue planet refers to planet Earth
Answer: Moon
Gap: Moon is the satellite of Earth

Figure 5: Knowledge gap between the **question** and the **answer** using the **fact**. For some complex questions, the fact clarifies certain concepts in the question, (e.g., "blue planet"), leading to a reformulation of the question based on the fact (e.g., "What is the satellite of Earth?") which is captured by this gap.

Question	Fact	Span	Relation	Gap
Q: A light bulb turns on when it receives energy from **A:** gasoline	a light bulb converts *electrical energy* into light energy when it is turned on	electrical energy	provides^{-1}, enables^{-1}	(gasoline, provides, electrical energy)
Q: What makes the best wiring? **A:** Tungsten	wiring requires an *electrical conductor*	electrical conductor	isa^{-1}, madeof	(Tungsten, is an, electrical conductor)

Table 1: Examples from KGD dataset. Note that the knowledge gap is captured in the form of (Span, Relation, Answer) but not explicitly annotated. $^{-1}$ is used to indicate that the argument order should be flipped.

Fact-to-Answer gap annotation into **two steps**: key term identification and relation identification.

Question Subset. First, we identified valid question-fact pairs where the fact supports the correct answer (verified via crowdsourcing) but does not trivially lead to the answer (fact only overlaps with the correct answer). Second, we noticed that the Fact-to-Answer gaps were much noisier for longer answer options, where you could write multiple knowledge gaps or a single complex knowledge gap. So we created *OBQA-Short*, the subset of OpenBookQA where answer choices have at most two non-stopword tokens. This contains over 50% of the original questions and is also the target set of our approach.

Two-step Gap Identification. Starting with the above pairs of questions with valid partial knowledge, the second task is to author facts that close the Fact-to-Answer **knowledge gap**. Again, initial iterations of the task resulted in poor quality, with workers often writing noisy facts that restated part of the provided fact or directly connect the question to the answer (skipping over the provided fact[4]). We noticed that the core fact often contains a key span that hints at the final answer. So we broke the task into two steps: (1) identify key terms (preferably a span) in the core fact that could answer the question, and (2) identify one or more relations[5] that hold between the key terms and the correct answer choice but not the incorrect choices. Table 1 shows example annotations of the gaps obtained through this process.

Knowledge Gap Dataset: KGD

Our Knowledge Gap Dataset (KGD) contains key span and relation label annotations to capture knowledge gaps. To reduce noise, we only use knowledge gap annotations where at least two of three workers found a contiguous span from the core fact and a relation from our list. The final

[4]This was also noticed by the original authors of OpenBookQA dataset (Mihaylov et al., 2018).

[5]Workers preferably chose from a selected list of nine most common relations: {causes, definedAs, enables, isa, located in, made of, part of, provides, synonym of} and their inverses (except synonymy). These relations have also been found to be useful by prior approaches for science QA (Clark et al., 2014; Khashabi et al., 2016; Jansen et al., 2016, 2018).

	Train	Dev	Test
Total #questions	1151	117	121
Total #question-facts	1531	157	165
Avg. # spans	1.43	1.46	1.45
Avg. # relations	3.31	2.45	2.45

Table 2: Statistics of the train/dev/test split of the KGD dataset. The #question-fact pairs is higher than #questions as some questions may be supported by multiple facts. The average statistic computes the average number of unique spans and relations per question-fact pair.

dataset contains examples of the form {question, fact, spans, relations}, where each span is a substring of the input fact, and relations are the set of valid relations between the span and the correct answer (examples in Table 1 and stats in Table 2).

4 Knowledge-Gap Guided QA: GapQA

We first introduce the notation used to describe our QA system. For each question q and fact f, the selected span is given by s and the set of valid relations between this span and the correct choice is given by r. Borrowing notation from OpenBookQA, we refer to the question without the answer choices c as the stem q_s, i.e., $q = q_s\ c$. We use $\hat{s}$ to indicate the predicted span and $\hat{r}$ for the predicted relations. We use q_m and f_m to represent the tokens in the question stem and fact respectively. Following the Turk task, our model first identifies the key span from the fact and then identifies the relation using retrieved knowledge.

4.1 Key Span Identification Model

Since the span selected from the fact often tends to be the answer to the question (c.f. Table 1), we can use a reading comprehension model to identify this span. The fact serves as the input passage and the question stem as the input question to the reading comprehension model. We used the Bi-Directional Attention Flow (BiDAF) model (Seo et al., 2017), an attention-based span prediction model designed for the SQuAD RC dataset (Rajpurkar et al., 2016). We refer the reader to the original paper for details about the model.

4.2 Knowledge Retrieval Module

Given the predicted span, we retrieve knowledge from two sources: triples from ConceptNet (Speer et al., 2017) and sentences from ARC corpus (Clark et al., 2018). ConceptNet contain (subject, relation, object) triples with relations such as */r/IsA*, */r/PartOf* that closely align with the relations in our gaps. Since ConceptNet can be incomplete or vague (e.g. */r/RelatedTo* relation), we also use the ARC corpus of 14M science-relevant sentences to improve our recall.

Tuple Search. To find relevant tuples connecting the predicted span $\hat{s}$ to the answer choice c_i, we select tuples where at least one token[6] in the subject matches $\hat{s}$ and at least one token in the object matches c_i (or vice versa). We then score each tuple t using the Jaccard score[7] and pick the top k tuples for each c_i ($k = 5$ in our experiments).

Text Search. To find the relevant sentences for $\hat{s}$ and c_i, we used ElasticSearch[8] with the query: $\hat{s}$ + c_i (refer to Appendix D for more details). Similar to ConceptNet, we pick top 5 sentences for each answer choice. To ensure a consistent formatting of all knowledge sources, we convert the tuples into sentences using few hand-defined rules(described in Appendix C). Finally all the retrieved sentences are combined to produce the input KB for the model, K.

4.3 Question Answering Model

The question answering model takes as input the question q_s, answer choices c, fact f, predicted span, $\hat{s}$ and retrieved knowledge K. We use 300-dimensional 840B GloVe embeddings (Pennington et al., 2014) to embed each word in the inputs. We use a Bi-LSTM with 100-dimensional hidden states to compute the contextual encodings for each string, e.g., $\mathcal{E}_f \in \mathbb{R}^{f_m \times h}$. The question answering model selects the right answer using two components: (1) **Fact Relevance** module (2) **Relation Prediction** module.

Fact Relevance. This module is motivated by the intuition that a relevant fact will often capture a relation between concepts that align with the question and the correct answer (the cyan and magenta regions in Figure 2). To deal with the gaps between these concepts, this module relies purely on word embeddings while the next module will focus on using external knowledge.

We compute a question-weighted and answer-weighted representation of the fact to capture the part of the fact that links to the question and answer respectively. We compose these fact-based

[6] We use lower-cased, stemmed, non-stopword tokens.

[7] score(t) = jacc(tokens(t), tokens($\hat{s}$ + c_i)) where jacc(w1, w2) = $\frac{w1 \cap w2}{w1 \cup w2}$

[8] https://www.elastic.co/products/elasticsearch

representations to then identify how well the answer choice is supported by the fact.

To calculate the question-weighted fact representation, we first identify facts words with a high similarity to some question word ($\mathcal{V}_{q_s}(f)$) using the attention weights: $\mathcal{A}_{q_s,f} = \mathcal{E}_{q_s} \cdot \mathcal{E}_f \in \mathbb{R}^{q_m \times f_m}$

$$\mathcal{V}_{q_s}(f) = \text{softmax}_{f_m}\left(\max_{q_m} \mathcal{A}_{q_s,f}\right) \in \mathbb{R}^{1 \times f_m}$$

The final attention weights are similar to the Query-to-Context attention weights in BiDAF. The final question-weighted representation is:

$$\mathcal{S}_{q_s}(f) = \mathcal{V}_{q_s}(f) \cdot \mathcal{E}_f \in \mathbb{R}^{1 \times h} \quad (1)$$

We similarly compute the choice-weighted representation of fact as $\mathcal{S}_{c_i}(f)$. We compose these two representations by averaging[9] these two vectors $\mathcal{S}_{q_s c_i}(f) = (\mathcal{S}_{q_s}(f) + \mathcal{S}_{c_i}(f))/2$. We finally score the answer choice by comparing this representation with the aggregate fact representation, obtained by averaging too, as:

$$\text{score}_f(c_i) = \text{FF}\left(\bigotimes(\mathcal{S}_{q_s c_i}(f), \text{avg}(\mathcal{E}_f)\right)$$

where $\bigotimes(x, y) = [x - y; x * y] \in \mathbb{R}^{1 \times 2h}$ and FF is a feedforward neural network that outputs a scalar score for each answer choice.

Filling the Gap: Relation Prediction. The relation prediction module uses the retrieved knowledge to focus on the Fact-to-Answer gap by first predicting the relation between $\hat{s}$ and c_i and then compose it with the fact to score the choice. We first compute the span and choice weighted representation($\mathbb{R}^{1 \times h}$) for each sentence k_j in K using the same operations as above:

$$\mathcal{S}_{\hat{s}}(k_j) = \mathcal{V}_{\hat{s}}(k_j) \cdot \mathcal{E}_{k_j} \; ; \; \mathcal{S}_{c_i}(k_j) = \mathcal{V}_{c_i}(k_j) \cdot \mathcal{E}_{k_j}$$

These representations capture the contextual embeddings of the words in the k_j that most closely resemble words in $\hat{s}$ and c_i respectively. We predict the kb-based relation between them based on the composition of these representations :

$$\mathcal{R}_j(\hat{s}, c_i) = \text{FF}\left(\bigotimes(\mathcal{S}_{\hat{s}}(k_j), \mathcal{S}_{c_i}(k_j))\right) \in \mathbb{R}^{1 \times h}$$

We pool the relation representations from all the KB facts into a single prediction by averaging, i.e. $\mathcal{R}(\hat{s}, c_i) = \text{avg}_j \; \mathcal{R}_j(\hat{s}, c_i)$.

[9]We found this simple composition function performed better than other composition operations.

Relation Prediction Score. We first identify the potential relations that can be composed with the fact, given the question, e.g., in Figure 1, we can compose the fact with (steel spoon; *made of*; metal) relation but not (metal; *made of*; ions). We compose an aggregate representation of the question and fact encoding to capture this information:

$$\mathcal{D}(q_s, f) = \bigotimes(\max_{q_m} \mathcal{E}_{q_s}, \max_{f_m} \mathcal{E}_f) \in \mathbb{R}^{1 \times 2h}$$

We finally score the answer choice based on this representation and the relation representation:

$$\text{score}_r(c_i) = \text{FF}\left([\mathcal{D}(q_s, f); \mathcal{R}(\hat{s}, c_i)]\right)$$

The final score for each answer choice is computed by summing the fact relevance and relation prediction based scores i.e. $\text{score}(c_i) = \text{score}_f(c_i) + \text{score}_r(c_i)$. The final architecture of our QA model is shown in Figure 6.

4.4 Model Training

We use cross-entropy loss between the predicted answer scores $\hat{c}$ and the gold answer choice $\bar{c}$. Since we also have labels on the true relations between the gold span and the correct answer choice, we introduce an auxiliary loss to ensure the predicted relation $\mathcal{R}$ corresponds to the true relation between s and c_i. We use a single-layer feedforward network to project $\mathcal{R}(s, c_i)$ into a vector $\hat{r}_i \in \mathbb{R}^{1 \times l}$ where l is the number of relations. Since multiple relations can be valid, we create an n-hot vector representation $\bar{r} \in \mathbb{R}^{1 \times l}$ where $\bar{r}[k] = 1$ if r_k is a valid relation.

We use binary cross-entropy loss between the $\hat{r}_i$ and r for the correct answer choice. For the incorrect answer choice, we do not know if any of the unselected relations(i.e. where $r[k] = 0$) hold. But we do know that the relations selected by Turkers for the correct answer choice should not hold for the incorrect answer choice. To capture this, we compute the binary cross entropy loss between $\hat{r}_i$ and $1 - r$ for the incorrect answer choices but ignore the unselected relations.

Finally, the loss for each example, assuming c_i is the correct answer, is given as loss $= ce(\hat{c}, \bar{c}) + \lambda \cdot \big(bce(\hat{r}_i, \bar{r}) + \sum_{j \neq i} mbce(\hat{r}_j, 1 - \bar{r}, \bar{r})\big)$, where ce is cross-entropy loss, bce is binary cross-entropy loss, and $mbce$ is masked binary cross entropy loss, where unselected relations are masked.

We further augment the training data with questions in the OBQA-Short dataset using the predicted spans and ignoring the relation loss. Also,

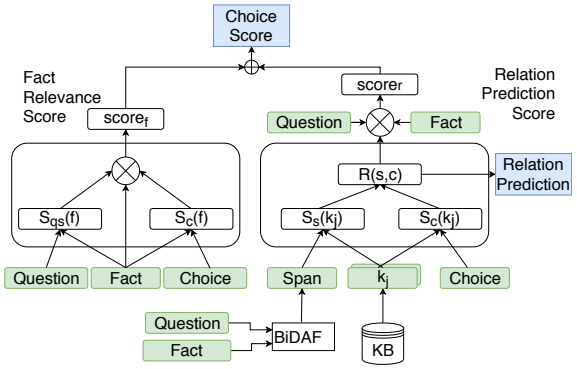

Figure 6: Overall architecture of the KGG question-answering model for each answer choice. The green nodes are the input to the model and the blue nodes are the model outputs that the losses are computed against. The model uses BiDAF to predict the key spans and retrieves KB facts based on the span and the input choice.

we assume the labelled core fact in the OpenBookQA dataset provides the partial knowledge needed to answer these questions.

Implementation details and parameter settings are deferred to Appendix B. A sample visualization of the attentions and knowledge used in the model are provided in Figure 10 in the Appendix.

5 Experimental Results

We present results of our proposed model, GapQA, on two question sets: (a) those with short answers,[10] OBQA-Short (290 test questions), and (b) the complete set, OBQA-Full (500 test questions). As we mentioned before, OBQA-Short subset is likely to have Fact-to-Answer gaps that can be targeted by our approach and we therefore expect larger and more meaningful gains on this subset.

5.1 Key Span Identification

We begin by evaluating three training strategies for the key span identification model, using the annotated spans in KGD for training. As seen in Table 3, the BiDAF model trained on the SQuAD dataset (Rajpurkar et al., 2016) performs poorly on our task, likely due to the different question style in OpenBookQA. While training on KGD (from scratch) substantially improves accuracy, we observe that using KGD to fine-tune BiDAF pretrained on SQuAD results in the best F1 (78.55) and EM (63.99) scores on the Dev set. All subsequent experiments use this fine-tuned model.

[10] Answers with at most two non-stopword tokens.

Training Data	Dev F1	Dev EM
SQuAD	54.67	41.40
KGD	72.99	58.60
SQuAD + KGD (tuning)	**78.55**	**63.69**

Table 3: BiDAF model performance on the span prediction task, under different choices of training data

5.2 OpenBookQA Results

We compare with three previous state-of-the-art **models** reported by Mihaylov et al. (2018). Two of these are Knowledge-free models (also referred to as No Context Baselines (Chen and Durrett, 2019)): (a) Question-to-Choice (**Q2Choice**) computes attention between the question and the answer choice, (b) **ESIM + ELMo**, uses ESIM (Chen et al., 2017) with ELMo (Peters et al., 2018) embeddings to compute question-choice entailment. The third is Knowledge Enhanced Reader (**KER**), which uses the core fact (f) and knowledge retrieved from ConceptNet to compute cross-attentions between the question, knowledge, and answer choices.

For **knowledge**, we consider four sources: (1) **ConceptNet (CN)**, the English subset of ConceptNet v5.6.0 tuples;[11] (2) **WordNet**, the WordNet subset of ConceptNet used by Mihaylov et al. (2018); (3) **OMCS**, the Open Mind Common Sense subset of ConceptNet used by Mihaylov et al. (2018); and (4) **ARC**, with 14M science-relevant sentences from the AI2 Reasoning Challenge dataset (Clark et al., 2018).

Following OpenBookQA, we train each model five times using different random seeds, and report the average score and standard deviation (without Bessel's correction) on the test set. For simplicity and consistency with prior work, we report one std. dev. from the mean using the $\mu \pm \sigma$ notation.

We train our model on the combined KGD and OBQA-Short question set with full supervision on examples in KGD and only QA supervision (with predicted spans) on questions in OBQA-Short. We train the baseline approaches on the entire question set as they have worse accuracies on both the sets when trained on the OBQA-Short subset. We do not use our annotations for any of the test evaluations. We use the core fact provided by the original dataset and use the predicted spans from the fine-tuned BiDAF model.

We present the test accuracies on the two ques-

[11] https://github.com/commonsense/conceptnet5/wiki

Model	OBQA-Short	OBQA-Full
Q2Choice	47.10 ± 1.5	49.64 ± 1.3
ESIM + ELMo	45.93 ± 2.6	49.96 ± 2.5
KER (only f)	57.93 ± 1.4	55.80 ± 1.8
KER (f + WordNet)	54.83 ± 2.5	55.84 ± 1.7
KER (f + OMCS)	49.65 ± 2.0	52.50 ± 0.8
GapQA (f + KB) [Ours]	**64.41 ± 1.8***	**59.40 ± 1.3***

Table 4: Test accuracy on the the OBQA-Short subset and OBQA-Full dataset assuming core fact is given. $*$ denotes the results are statistically significantly better than all the baselines (p≤0.05, based on Wilson score intervals (Wilson, 1927)).

Knowledge Source	Model	OBQA-Short
f + WordNet	KER	54.83 ± 2.5
	GapQA	**60.69 ± 1.1***
f + OMCS	KER	49.65 ± 2.0
	GapQA	**60.90 ± 2.4***
f + CN + ARC	GapQA	**64.41 ± 1.8**

Table 5: Test accuracy on the OBQA-Short subset with different sources of knowledge. $*$ denotes the results are statistically significantly better than the corresponding KER result (p≤0.05, based on Wilson score intervals (Wilson, 1927)).

Model	OBQA-Short	Δ
GapQA	**64.41 ± 1.8**	—
No Annotations	58.90 ± 1.9	5.51
Heuristic Span Anns.	61.38 ± 1.5	3.03
No Relation Score	60.48 ± 1.1	3.93
No Spans (Model)	62.14 ± 2.1	2.27
No Spans (IR)	61.79 ± 1.0	2.62

Table 6: Average accuracy of various ablations, showing that each component of GapQA has an important role. No Annotations = Ignore Span & Relation Annotations, Heuristic Span Anns = Heuristically predict span annotations (no human annotations), No Relation Score = Ignore the relation-based score ($score_r$), No Spans (Model) = Ignore the span (use entire fact) to compute span-weighted representations, No Spans (IR) = Ignore the span (use entire fact) for retrieval.

tion sets in Table 4. On the targeted OBQA-Short subset, our proposed GapQA improves statistically significantly over the partial knowledge baselines by 6.5% to 14.4%. Even though the full OpenBookQA dataset contains a wider variety of questions not targeted by GapQA, we still see an improvement of 3+% relative to prior approaches.

It is worth noting that recent large-scale language models (LMs) (Devlin et al., 2019; Radford et al., 2018) have now been applied on this task, leading to improved state-of-the-art results (Sun et al., 2018; Banerjee et al., 2019; Pan et al., 2019). However, our knowledge-gap guided approach to QA is orthogonal to the underlying model. Combining these new LMs with our approach is left to future work.

Effect of input knowledge. Since the baseline models use different knowledge sources as input, we evaluate the performance of our model using the same knowledge as the baselines.[12] Even when our model is given the same knowledge, we see an improvement by 5.9% and 11.3% given only WordNet and OMCS knowledge respectively. This shows that we can use the available knowledge, even if limited, more effectively than previous methods. When provided with the full ConceptNet knowledge and large-scale text corpora, our model is able to exploit this additional knowledge and improve further by 4%.

5.3 Ablations

We next evaluate key aspects of our model in an ablation study, with average accuracies in Table 6.

No Annotations (No Anns): We ignore all collected annotations (span, relation, and fact) for training the model. We use the BiDAF(SQuAD) model for span prediction, and only the question answering loss for the QA model trained on the OBQA-Short subset.[13] Due to the noisy spans produced by the out-of-domain BiDAF model, this model performs worse that the full GapQA model by 5.5% (comparable performance to the KER models). This shows that our model does utilize the human annotations to improve on this task.

Heuristic Span Annotations: We next ask whether some of the above loss in accuracy can be recovered by heuristically producing the spans for training—a cost-effective alternative to human annotations. We find the longest subsequence of tokens (ignoring stop words) in f that is not mentioned in q and assume this span (including the intermediate stop words) to be the key term. To prevent noisy key terms, we only consider a subset of questions where 60% of the non-stopword stemmed tokens in f are covered by q. We fine-tune the BiDAF(SQuAD) model on this subset and then use it to predict the spans on the full set.[14]

[12] The baselines do not scale to large scale corpora and so can not be evaluated against our knowledge sources.

[13] Model still predicts the latent relation representation.

[14] We use the questions from the KGD + OBQA-Short set. Note we are only evaluating the impact of heuristic spans compared to human-authored spans, but assume that we have

Question	Fact	Predicted Answer	Reason
What vehicle would you use to travel on the majority of the surface of the planet on which we live? (A) Bike (B) **Boat** (C) Train (D) Car	oceans cover 70% of the surface of the earth	Bike	Predicted the **wrong span** "70%".
What contains moons? (A) ships (B)**space mass** (C) people (D) plants	the solar system contains the moon	ships	**Scores the relation for the incorrect answer higher** because of the facts connecting "systems" and "ships".
Cocoon creation occurs (A) **after the caterpillar stage** (B) after the chrysalis stage (C) after the eggs are laid (D) after the cocoon emerging stage	the cocoons being created occurs during the pupa stage in a life cycle	after the chrysalis stage	Does not model the **complex relation** (temporal ordering) between the key span: "pupa stage" and "caterpillar stage". Instead it predicts "chrysalis" due to the synonymy with "pupa".

Table 7: Sample errors made by the GapQA on questions from the OBQA-Short dataset. The correct answers are marked in **bold** within the question.

We train GapQA model on this dataset without any relation labels (and associated loss). This simple heuristic leads to a 3% drop compared to human annotations, but still out-performs previous approaches on this dataset, showing the value of the gap-based QA approach.

No Relation Score: We ignore the entire relation-based score ($score_r$) in the model and only rely on the fact-relevance score. The drop in score by 3.9% shows that the fact alone is not sufficient to answer the question using our model.

No Spans (Model): We ignore the spans in the model, i.e., we use the entire fact to compute the span-based representation $\mathcal{S}_{\hat{s}}(k_j)$. In effect, the model is predicting the gap between the entire fact and answer choice.[15] We see a drop of $\sim$2%, showing the value of spans for gap prediction.

No Spans (IR): Ignoring the span for retrieval, the knowledge is retrieved based on the entire fact (full GapQA model is used). The drop in accuracy by 2.6% shows the value of targeted knowledge-gap based retrieval.

5.4 Error Analysis

We further analyzed the performance of GapQA on 40 incorrectly answered questions from the dev set in the OBQA-Short dataset. Table 7 shows a few error examples. There were three main classes of errors:

Incorrect predicted spans (25%) often due to complex language in the fact or the Question-to-Fact gap needed to accurately identify the span.

Incorrect relation scores (55%) due to distracting facts for the incorrect answer or not finding relevant facts for the correct answer, leading to an incorrect answer scoring higher.

Out-of-scope gap relations (20%) where the knowledge gap relations are not handled by our model such as temporal relations or negations (e.g., is *not* made of).

Future work in expanding the dataset, incorporating additional relations, and better retrieval could mitigate these errors.

6 Conclusion

We focus on the task of question answering under partial knowledge: a novel task that lies in-between open-domain QA and reading comprehension. We identify classes of knowledge gaps when reasoning under partial knowledge and collect a dataset targeting one common class of knowledge gaps. We demonstrate that identifying the knowledge gap first and then reasoning by filling this gap outperforms previous approaches on the OpenBookQA task, with and even without additional missing fact annotation. This work opens up the possibility of focusing on other kinds of knowledge gaps and extending this approach to other datasets and tasks (e.g., span prediction).

Acknowledgements

We thank Amazon Mechanical Turk workers for their help with annotation, and the reviewers for their invaluable feedback. Computations on beaker.org were supported in part by credits from Google Cloud.

good quality partial context as provided in KGD.

[15] Retrieval is still based on the span and we ignore the relation prediction loss.

References

Gabor Angeli, Neha Nayak, and Christopher D. Manning. 2016. Combining natural logic and shallow reasoning for question answering. In *Association for Computational Linguistics (ACL)*.

Pratyay Banerjee, Kuntal Kumar Pal, Arindam Mitra, and Chitta Baral. 2019. Careful selection of knowledge to solve open book question answering. In *ACL*.

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 *SIGMOD*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. In *NIPS*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *NAACL*.

Jifan Chen and Greg Durrett. 2019. Understanding dataset design choices for multi-hop reasoning. In *NAACL-HLT*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *ACL*, pages 1657–1668.

Peter Clark, Niranjan Balasubramanian, Sumithra Bhakthavatsalam, Kevin Humphreys, Jesse Kinkead, Ashish Sabharwal, and Oyvind Tafjord. 2014. Automatic construction of inference-supporting knowledge bases. In *AKBC Workshop*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter D Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *AAAI*, pages 2580–2586.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. In *NAACL*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *ICML*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Diana Inkpen, Xiao-Dan Zhu, Zhen-Hua Ling, Qian Chen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *ACL*.

Peter Jansen, Niranjan Balasubramanian, Mihai Surdeanu, and Peter Clark. 2016. What's in an explanation? Characterizing knowledge and inference requirements for elementary science exams. In *COLING*.

Peter Jansen, Rebecca Sharp, Mihai Surdeanu, and Peter Clark. 2017. Framing qa as building and ranking intersentence answer justifications. *Computational Linguistics*, 43(2):407–449.

Peter A. Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton T. Morrison. 2018. WorldTree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. In *LREC*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *ACL*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL*.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *Proceedings of IJCAI*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *EMNLP*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable Reader: Enhancing Cloze-Style Reading Comprehension with External Commonsense Knowledge. In *ACL*, pages 821–832.

Xiaoman Pan, Kai Sun, Dian Yu, Heng Ji, and Dong Yu. 2019. Improving question answering with external knowledge. *CoRR*, abs/1902.00993.

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

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Michael Petrochuk and Luke S. Zettlemoyer. 2018. Simplequestions nearly solved: A new upperbound and baseline approach. In *EMNLP*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *[Online]*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, pages 193–203.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *CoRR*, abs/1809.02040.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *AAAI*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2018. Improving machine reading comprehension with general reading strategies. *CoRR*, abs/1810.13441.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Workshop on Representation Learning for NLP*.

Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural nlu systems. *arXiv preprint arXiv:1706.02596*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*, 6:287–302.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.

Edwin B. Wilson. 1927. Probable inference, the law of succession, and statistical inference. *JASA*, 22(158):209–212.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

A Annotating Gaps: Data Collection

We first identify relevant facts for questions and then collect annotations for fact-answer gap, given the relevant fact. However, straightforward approaches to annotate all questions led to noisy labels. To improve annotation quality, we identified question subsets most suitable for this task and split the fact-answer gap annotation into two steps.

Fact Relevance. The OpenBookQA dataset provides the core science fact used to create the question. However, in 20% of the cases, while the core science fact inspired the question, it is not needed to answer the question (Mihaylov et al., 2018). We also noticed that often multiple facts from the open book can be relevant for a question. So we first create an annotation task to identify the relevant facts from a set of retrieved facts. Also to ensure that there is a gap between the fact and the correct answer, we select facts that have no word overlap with the correct choice or have overlap with multiple answer choices. This ensures that the fact alone can not be trivially used to answer the question.

We ask Turkers to annotate these retrieved facts as (1) are they *relevant* to the question and (2) if relevant, do they point to a *unique* answer. We introduced the second category after noticing that some generic facts can be relevant but not point to a specific answer making identifying the knowledge gap impossible. E.g. The fact: "evaporation is a stage in the water cycle process" only eliminates one answer option from "The only stage of the water cycle process that is nonexistent is (A) evaporation (B) evaluation (C) precipitation (D) condensation". For each question, we selected facts that were marked as relevant and unique by at least two out of three turkers.

Knowledge Gap. In the second round of data collection, we asked Turkers to write the facts connecting the relevant fact to the correct answer choice. We restricted this task to Masters level Turkers with 95% approval rating and 5000 approved hits. However, we noticed that crowdsource workers would often re-state part of the knowledge mentioned in the original fact or directly connect the question to the answer. This issue was also mentioned by the authors of OpenBookQA who also noticed that the additional facts were "*noisy (incomplete, over-complete, or only distantly related)*" (Mihaylov et al., 2018). E.g. for the question: "In the desert, a hawk may enjoy an occasional (A) coyote (B) reptile (C) bat (D) scorpion" and core fact: "hawks eat lizards", one of the turk-authored additional fact: "Hawks hunt reptiles which live in the desert" is sufficient to answer the question on its own.

We also noticed that questions with long answer choices often have multiple fact-answer gaps leading to complex annotations, e.g. "tracking time" *helps with* "measuring how many marshmallows I can eat in 10 minutes". Collecting knowledge gaps for such questions and common-sense knowledge to capture these gaps are interesting directions of future research. We instead focus on questions where the answer choices have at most two non-stopword tokens. We refer to this subset of questions in OpenBookQA as *OBQA-Short*, which still forms more than 50% of the OpenBookQA set. This subset also forms the target question set of our approach.

Further to simplify this task, we broke the task of identifying the required knowledge into two steps (shown in Figure 7 in Appendix): (1) identify key terms in the core fact that could answer the question, and (2) identify the relationship between these terms and the correct answer choice. For key terms, we asked the Turkers to select spans from the core fact itself, to the extent possible. For the relation identification, we provided a list of relations and asked them to select all the relations that hold between the key term and the correct choice but do not hold for the incorrect answer choices. Based on our analysis, we picked nine most common relations: {causes, definedAs, enables, isa, located in, made of, part of, provides, synonym of} and their inverses (except synonymy).[16] If none of these relations were valid, they were allowed to enter the relation in a text box.

We note that the goal of this effort was to collect supervision for a subset of questions to guide the model and show the value of minimal annotation on this task. We believe our approach can be useful to collect annotations on other question sets as well, or can be used to create a challenge dataset for this sub-task. Moreover, the process of collecting this data revealed potential issues with collecting annotations for knowledge gaps and also inspired the design of our two-step QA model.

[16] These relations were also found to be important by prior approaches (Clark et al., 2014; Khashabi et al., 2016; Jansen et al., 2016, 2018) in the science domain.

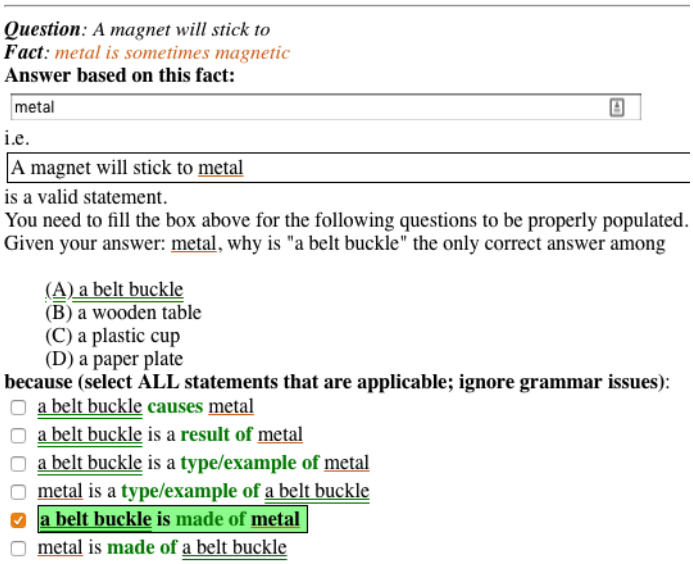

Figure 7: Interface provided to Turkers to annotate the missing fact. Entering the answer span from the fact, metal, in this example, automatically populates the interface with appropriate statements. The valid statements are selected by Turkers and capture the knowledge gap.

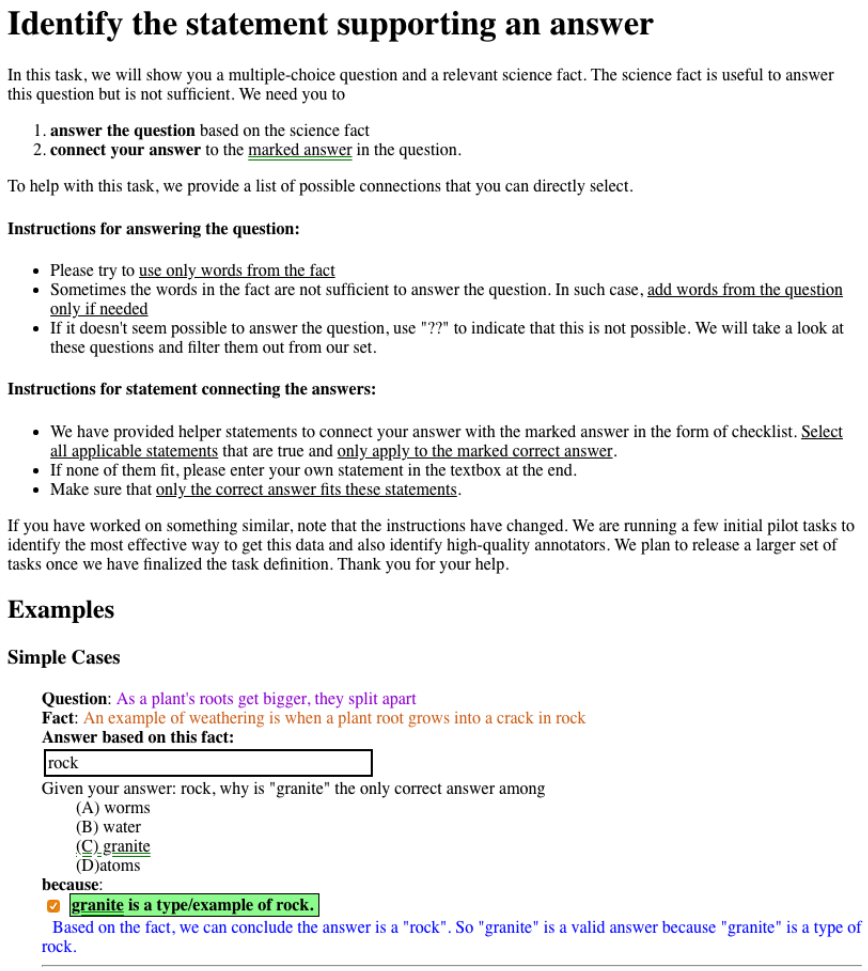

Figure 8: Basic Instructions for the task

Hard Cases

Writing your own statement: If none of the provided statements apply, write your own statment.

Question: What happens when a hemisphere is tilted away from the sun?
Fact: winter is when a hemisphere is tilted away from the sun.
Answer based on this fact:
winter
Given your answer, why is "cools" the only correct answer ?
(A) cools
(B) nothing
(C) heats
(D) warms
because:
If none of the above apply, write your own statement here using cools and winter
cools happens during winter ⟵
Since none of the pre-defined list of helper phrases fit, write your own.

Writing your own answer: If you can't find the right answer as a part of the fact, write your own answer.

Question: An animal living in an environment lacking in food resources
Fact: an animal requires enough nutrients to maintain good health
Answer based on this fact:
not maintain good health ⟵
Since the answer is the opposite of the "maintain good health" in the fact, we need to add a "not" to the answer.
Given your answer, why is "will be in poor shape" the only correct answer ?
(A) will be in poor shape
(B) will be thriving and lively
(C) will be switching to a new diet
(D) will hibernate until more food comes along
because: ☑ **will be in poor shape is synonymous with will maintain bad health.**

Adding words from the question: Sometimes words from the question need to be added to the statement to ensure that only the correct answer fits the statement.

Question: Which would be the result of the breeding of two wolves?
Fact: reproduction produces offspring
Answer based on this fact:
offspring
Given your answer, why is "wolf pups" the only correct answer ?
(A) kittens
(B) wolf pups
(C) fox pups
(D) dog pups
because:
If none of the above apply, write your own statement here using wolf pups and offspring
wolf pups are offsprings of wolves. ⟵
Since all the answers are offsprings, we additionally specify that the correct answer should be offspring of wolves.

Figure 9: Instructions for complex examples

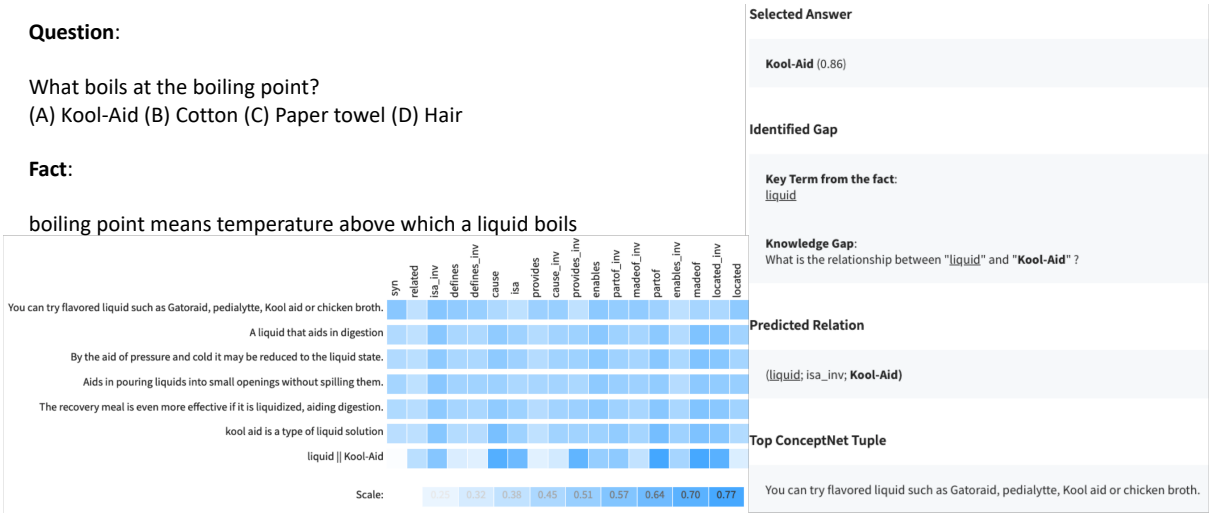

Figure 10: Visualization of the models behavior with the predicted span, top predicted relation, and the top fact used by model. The heat map shows the confidence of the model for all the relations for each input sentence (first five) and ConceptNet sentencized tuple (last but one) and the back-off tuple (last one) to capture the knowledge in the embeddings.

B Implementation Details

We implement all our models in Pytorch (Paszke et al., 2017) using the AllenNLP (Gardner et al., 2017) toolkit. We also used the AllenNLP implementation of the BiDAF model for span prediction. We use 300D 840B Glove (Pennington et al., 2014) embeddings and use 200 dimensional hidden representations for the BiLSTM shared between all inputs (each direction uses 100 dimensional hidden vectors). We use 100 dimensional representations for the relation prediction, $\mathcal{R}_j$. Each feedforward network, FF is a 2-layer network with relu activation, 0.5 dropout (Srivastava et al., 2014), 200 hidden dimensions on the first layer and no dropout on the output layer with linear activation. We use a variational dropout (Gal and Ghahramani, 2016) of 0.2 in all the BiLSTMs. The relation prediction loss is scaled by $\lambda = 1$. We used the Adam (Kingma and Ba, 2015) optimization with initial $lr = 0.001$ and a learning rate scheduler that halves the learning rate after 5 epochs of no change in QA accuracy. We tuned the hyper-parameters and performed early stopping based on question answering accuracy on the validation set. Specifically, we considered {50, 100, 200} dimensional representations, $\lambda \in \{0.1, 1, 10\}$, retrieving {10, 20} knowledge tuples and {[x - y; x*y], [x, y]} combination functions for $\bigotimes$ during the development of the model. The baseline models were developed for this dataset using hyper-parameter tuning; we do not perform any additional tuning. Our model code and pre-trained models are available at https://github.com/allenai/missing-fact.

C ConceptNet sentences

Given a tuple $t = (s, v, o)$, the sentence form is generated as "s is split(v) o" where split(v) splits the ConceptNet relation v into a phrase based on its camel-case notation. For example, (belt buckle, /r/MadeOf, metal) would be converted into "belt buckle is made of metal".

D Text retrieval

For each span $\hat{s}$ and answer choice c_i, we query an ElasticSearch [17] index on the input text corpus with the "$\hat{s}$ + c_i" as the query. We also require the matched sentence must contain both the span and the answer choice. We filter long sentences (>300 characters), sentences with negation and noisy sentences[18] from the retrieved sentences.

[17] https://www.elastic.co/

[18] Sentences are considered clean if they contain alphanumeric characters with standard punctuation, start with an alphabet or a number, are single sentence and only uses hyphens in hyphenated word pairs