Repurposing Entailment for Multi-Hop Question Answering Tasks

Harsh Trivedi[♣], **Heeyoung Kwon**[♣], **Tushar Khot**[♠], **Ashish Sabharwal**[♠], **Niranjan Balasubramanian**[♣]

[♣] Stony Brook University, Stony Brook, U.S.A.
{hjtrivedi,heekwon,niranjan}@cs.stonybrook.edu
[♠] Allen Institute for Artificial Intelligence, Seattle, U.S.A.
{tushark,ashishs}@allenai.org

Abstract

Question Answering (QA) naturally reduces to an entailment problem, namely, verifying whether some text entails the answer to a question. However, for multi-hop QA tasks, which require reasoning with *multiple* sentences, it remains unclear how best to utilize entailment models pre-trained on large scale datasets such as SNLI, which are based on sentence pairs.

We introduce Multee, a general architecture that can effectively use entailment models for multi-hop QA tasks. Multee uses (i) a local module that helps locate important sentences, thereby avoiding distracting information, and (ii) a global module that aggregates information by effectively incorporating importance weights. Importantly, we show that both modules can use entailment functions pre-trained on a large scale NLI datasets. We evaluate performance on MultiRC and OpenBookQA, two multihop QA datasets. When using an entailment function pre-trained on NLI datasets, Multee outperforms QA models trained only on the target QA datasets and the OpenAI transformer models.

1 Introduction

How can we effectively use textual entailment models for question answering? Previous attempts at this have resulted in limited success (Harabagiu and Hickl, 2006; Sacaleanu et al., 2008; Clark et al., 2012). With recent large scale entailment datasets (Bowman et al., 2015; Williams et al., 2018; Khot et al., 2018) pushing entailment models to high accuracies (Chen et al., 2017; Parikh et al., 2016; Wang et al., 2017), we re-visit this challenge and propose a novel method for re-purposing neural entailment models for QA.

A key difficulty in using entailment models for QA turns out to be the mismatch between the inputs to the two tasks: large-scale entailment datasets are typically framed at a *sentence level*, whereas question answering requires verifying whether *multiple sentences*, taken together as a premise, entail a hypothesis.

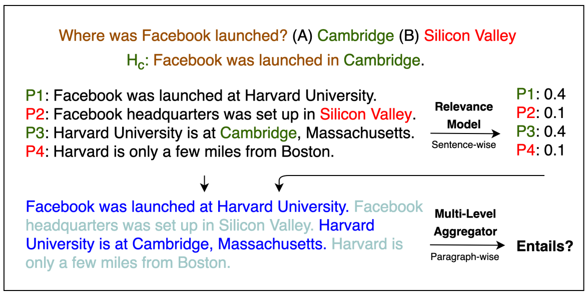

Where was Facebook launched? (A) Cambridge (B) Silicon Valley
H_c: Facebook was launched in Cambridge.
P1: Facebook was launched at Harvard University.
P2: Facebook headquarters was set up in Silicon Valley.
P3: Harvard University is at Cambridge, Massachusetts.
P4: Harvard is only a few miles from Boston.
Relevance Model
Sentence-wise
P1: 0.4
P2: 0.1
P3: 0.4
P4: 0.1
Facebook was launched at Harvard University. Facebook headquarters was set up in Silicon Valley. Harvard University is at Cambridge, Massachusetts. Harvard is only a few miles from Boston.
Multi-Level Aggregator
Paragraph-wise
Entails?

Figure 1: An example illustrating the challenges in using sentence-level entailment model for multi-sentence reasoning needed for QA, and the high-level approach used in Multee.

There are two straightforward ways to address this mismatch: (1) aggregate independent entailment decisions over each premise sentence, or (2) make a single entailment decision after concatenating all premise sentences. Neither approach is fully satisfactory. To understand why, consider the set of premises in Figure 1, which entail the hypothesis H_c. Specifically, the combined information in $P1$ and $P3$ entails H_c, which corresponds to the correct answer *Cambridge*. On one hand, aggregating independent decisions will fail because no individual premise entails H_C. On the other hand, simply concatenating premises to form a single paragraph will fail because distracting information in $P2$ and $P4$ can muddle useful information in $P1$ and $P3$. An effective approach, therefore, must recognize relevant sentences (i.e., avoid distracting ones) and compose their sentence-level information.

Our solution to this challenge is based on the observation that a sentence-level entailment function can be re-purposed for both recognizing relevant sentences, and for computing sentence-level representations. Both tasks require comparing information in a pair of texts, but the objectives of the comparison are different. This means we can take an entailment function that is trained for basic entailment (i.e., comparing information in texts), and adapt it to work for both recognizing relevance and computing representations. Thus, this architecture allows us to incorporate advances in entailment architectures and to leverage pre-trained models obtained using large scale entailment datasets.

To this end, we propose a general architecture that uses a (pre-trained) entailment function f_e for multi-sentence QA. Given a hypothesis statement H_{qa} representing a candidate answer, and the set of premise sentences $\{P_i\}$, our proposed architecture uses the same function f_e for two components: (a) a sentence relevance module that scores each P_i based on its potential relevance to H_{qa}, with the goal of weeding out distractors; and (b) a relevance-weighted aggregator that combines entailment information from multiple P_i.

Thus, we build effective entailment aware representations of larger contexts (i.e., multiple sentences) from those of small contexts (i.e., individual sentences). The main strength of our approach is that, unlike standard attention mechanisms, the aggregator module uses the attention scores from the relevance module at *multiple* levels of abstractions (e.g., multiple layers of a neural network) within f_e, using *join* operations that compose representations at each level. We refer to this **mult**i-**l**evel aggregation of **te**xtual **e**ntailment representations as Multee (pronounced multi).

Our implementation of Multee uses ESIM (Chen et al., 2017), a recent sentence-level entailment model, pre-trained on SNLI and MultiNLI datasets. We demonstrate its effectiveness on two challenging multi-sentence reasoning datasets: MultiRC (Khashabi et al., 2018) and OpenBookQA (Mihaylov et al., 2018). Multee using ELMo contextual embeddings (Peters et al., 2018) matches state-of-the-art results achieved with large transfomer-based models (Radford et al., 2018) that were trained on a sequence of large scale tasks (Sun et al., 2019). Ablation studies demonstrate that both relevance scoring and multi-level aggregation are valuable, and that pre-training on large entailment corpora is particularly helpful for OpenBookQA.

This work makes three main contributions: **(i)** A novel approach to use pre-trained entailment models for question answering. **(ii)** A model that incorporates local (sentence level) entailment decisions with global (document level) entailment decisions to effectively aggregate information for multi-hop QA task. **(iii)** An empirical evaluation that shows entailment based QA can achieve state-of-the-art performance on two challenging multi-hop QA datasets, OpenBookQA and MultiRC.

2 Question Answering using Entailment

Non-extractive question answering can be seen as a textual entailment problem, where we verify whether a hypothesis constructed out of a question and a candidate answer is entailed by the knowledge—a collection of sentences[1] in the source text. The probability of an answer A, given a question Q, can be modeled as the probability of a set of premises $\{P_i\}$ entailing a hypothesis statement H_{qa} constructed from Q and A:

$$\Pr[A \mid Q, \{P_i\}] \triangleq \Pr[\{P_i\} \vDash H_{qa}] \quad (1)$$

Here we use $\vDash$ to denote textual entailment. Given QA training data, we can then learn a model that approximates the entailment probability $\Pr[\{P_i\} \vDash H_{qa}]$.

Can one build an effective QA model g_e using an existing entailment model f_e that has been pre-trained on a large-scale entailment dataset? Figure 2 illustrates two straightforward ways of doing so, using f_e as a black-box function:

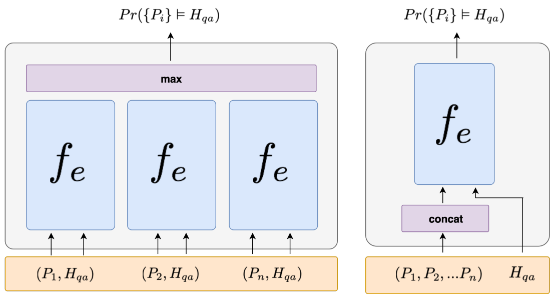


Figure 2: Black Box Applications of Textual Entailment Model for QA: Max and Concat models

(i) Aggregate Local Decisions (Max): Use f_e to check how much each sentence P_i entails H_{qa} on its own, and aggregate these local entailment decisions, for instance, using a max operation.

$$g_e(\{P_i\}, H_{qa}) = \max_i f_e(P_i, H_{qa}) \quad (2)$$

[1]This collection can be a *sequence* in the case of passage comprehension or a *list* of sentences, potentially from varied sources, in the case of QA over multiple documents.

(ii) Concatenate Premises (Concat): Combine the premise sentences in a sequence to form a single large passage P, and use f_e to check whether this passage as a whole entails the hypothesis H_{qa}, making a single entailment decision:

$$g_e(\{P_i\}, H_{qa}) = f_e(P, H_{qa}) \tag{3}$$

Our experiments reveal, however, that neither approach is an effective means of using pre-trained entailment models for QA (see Table 1). For the example in Figure 1, *Max* model would not be able to consider information from P1 and P3 together. Instead, it will pickup *Silicon Valley* as the answer since P2 is close to H_s, *"Facebook was launched in Silicon Valley"*. Similarly, *Concat* would also be muddled by distracting information in P2, which will weaken its confidence in answer *Cambridge*. Therefore, without careful guidance, simple aggregation can easily add distracting information into the premise representation, causing entailment to fail. This motivates the need for new, effective mechanisms for global reasoning over a collection of premises.

3 Our Approach: Multee

We propose a new entailment based QA model, Multee, with two components: (i) **a sentence relevance model**, which learns to focus on the relevant sentences, and (ii) **a multi-layer aggregator**, which uses an entailment model to obtain multiple layers of question-relevant representations for the premises and then composes them using the sentence-level scores from the relevance model. Finding relevant sentences is a form of local entailment between each premise and the answer hypothesis, whereas aggregating question-relevant representations is a form of global entailment between all premises and the answer hypothesis. This means, we can effectively re-purpose the same pre-trained entailment function f_e for both components. Figure 3 shows an architecture that uses multiple copies of f_e to achieve this.

3.1 Sentence Relevance Model

The goal of this module is to identify sentences in the paragraph that are important for the given hypothesis. As shown in Figure 1, this helps the global module aggregate relevant content while reducing the chances of picking up distracting information. A sentence is considered important if it contains information that is relevant to answering the question. In other words, the importance of a sentence can be modeled as its entailment probability, i.e., how well the sentence by itself supports the answer hypothesis. We can use a pre-trained entailment model to obtain this. The importance α_i of a sentence P_i can be modeled as:

$$\alpha_i = f_e(P_i, H_{qa}) \tag{4}$$

This can be further improved by modeling the sentence with its surrounding context. This is especially useful for passage-level QA, where the neighboring sentences provide useful context. Given a premise sentence P_i, the entailment function f_e computes a single hypothesis-aware representation x_i containing information in the premise that is relevant to entailing the answer hypothesis H_{qa}. This is essentially the output of last layer of neural function f_e before projecting it to logits. We denote this part of f_e that outputs last *vector* representation as f_{e_v} and full f_e that gives entailment *probability* as f_{e_p}.

We use these hypothesis-aware x_i vectors for each sentence as inputs to a BiLSTM producing a contextual representation c_i for each premise sentence P_i, which is then fed to a feedforward layer that predicts the sentence-level importance as:

$$\alpha_i = \mathrm{softmax}(W^T c_i + b) \tag{5}$$

The components for generating x_i are part of the original entailment function f_e and can be pre-trained on the entailment dataset. The BiLSTM to compute c_i and the parameters W and b for computing α_i are not part of the original entailment function and thus can only be trained on the target QA task. We perform this additional *contextualization* only when sentences form contiguous text. Additionally, for datasets such as MultiRC, where the relevant sentences have been marked, we introduce a loss term based on the true relevance label and predicted weights, α_i.

3.2 Multi-level Aggregation

The goal of this module is to aggregate representations from important sentences in order to make a global entailment decision. There are two key questions to answer: (1) how to combine the sentence-level information into a paragraph-level representation and (2) how to use the sentence relevance weights $\{\alpha_i\}$.

Most entailment models include many layers that transform the input premise and the hypothesis. A typical neural entailment stack includes en-

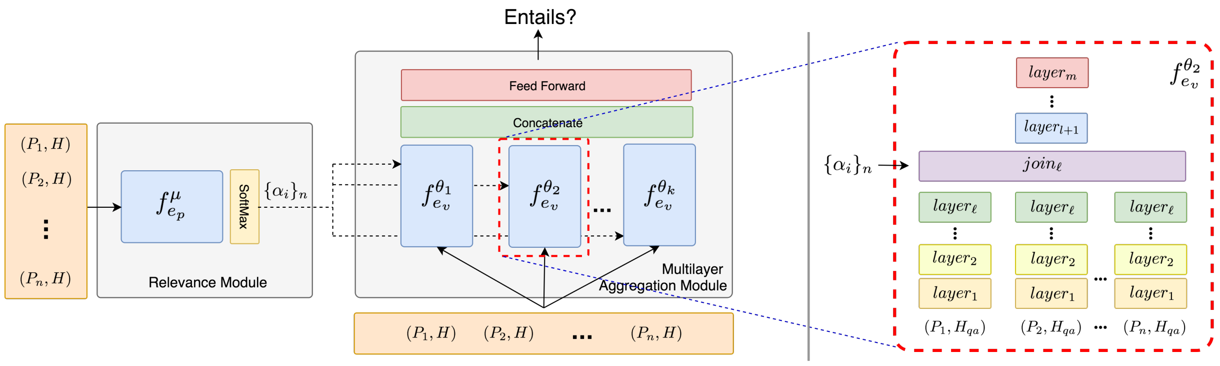


Figure 3: Multee overview: Multee includes two main components, a relevance module, and a multi-layer aggregator module. Both modules use pre-trained entailment functions (f_{e_p} and f_{e_v}). f_{e_p} is the full entailment model that gives entailment probability, and f_{e_v} is part of it excluding last projection to logits and softmax. The multi-level aggregator uses multiple copies of entailment function f_{e_v}, one for each sub-aggregator performing a *join* at a different layer. Right part of figure zooms in on one such sub-aggregator joining at layer ℓ.

coding layers that independently generate contextual representations of the premise and the hypothesis, followed by some cross-attention layer that yields relationships between the premise and hypothesis words, and additional layers that use this cross-attention to generate premise attended representations of the hypothesis and vice versa. The final layers are classification layers which determine entailment based on the representations from the previous layer. Each layer thus generates intermediate representation that captures different type of entailment related information. This presents us with a choice of multiple points for aggregation.

Figure 3 illustrates our approach for aggregating sentence-level representations into a single paragraph level representation. For each premise P_i in the passage, we first process the pair (P_i, H_{qa}) through the entailment stack (f_{e_v}) resulting in a set of intermediate representations $\{\tilde{X}_i^{\ell}\}$ for each layer ℓ. We can choose a particular layer ℓ to be the aggregation layer. We then compute a weighted combination of the sentence-level outputs at this layer $\{\tilde{X}_i^{\ell}\}$ to produce a passage-level representation $\tilde{Y}^{\ell}$. The weights for the sentences are obtained from the *Sentence Relevance* model. We refer to this as a *join* operation as shown in the Figure 3. Layers of the entailment function f_{e_v} that are below the *join* operate at a sentence-level, while layers above the join now operate over paragraph-wise representations. The final layer (i.e. the top most layer) of f_{e_v} thus gives us a vector representation of the entire passage. This type of join can be applied at multiple layers resulting in paragraph vectors that correspond to multiple levels of aggregation. We concatenate these paragraph vectors and pass them through a feed-forward network projecting them down to logits, that can be used to compute the final passage wide entailment probabilities.

3.2.1 Join Operations

Given a set of sentence-wise outputs from the lower layer $\{\tilde{X}_i\}$ and the corresponding sentence-relevance weights $\{\alpha_i\}$, the join operation combines them into a single passage-level representation $\tilde{Y}$, which can be directly consumed by the layer above it in the stack. The specifics of the join operation depends on the shape of the outputs from the lower layer, and the shape of the inputs expected by the layer after the join. Here we show four possible join operations, one for each layer. The ones defined for *Score Layer* and *Embedding Layer* can be reduced to black-box baselines, while we use the other two in Multee.

Score Layer: The score layer outputs the entailment probabilities $\{s_i\}$ for each premise to hypothesis independently, which need to be joined to one entailment score. One way to do this is to simply take a weighted maximum of the individual entailment probabilities. So we have $\tilde{X}_i = s_i \quad \forall i$ and $\tilde{Y} = \max_i \big(\alpha_i s_i\big)$. This reduces to black-box *Max* model (Equation 2) when using $\{\alpha_i\} = \mathbf{1}$.

Embedding Layer: The embedding layer outputs a sequence of embedded vectors of $[\bar{P}_i]$[2] one sequence for each premise P_i and another sequence of embedded vectors $[\bar{H}_{qa}]$ for the answer hypothesis H_{qa}. A join operation in this case scales each embedded vector in a premise by its relevance weight and concatenates them together to

[2]We use [.] to denote a sequence and $\bar{.}$ to denote a vector

form $[\bar{P}]$. $\bar{H}_{qa}$ is passed through unchanged.

$$\begin{aligned}\tilde{X}_i &= ([\bar{P}_i], [\bar{H}_{qa}]) \quad \forall i\\ [\bar{P}] &= [\alpha_1[\bar{P}_1]; \alpha_2[\bar{P}_2]; \ldots; \alpha_n[\bar{P}_n]]\\ \tilde{Y} &= \left([\bar{P}], [\bar{H}_{qa}]\right)\end{aligned}$$

For non-contextual word embeddings, this reduces to *Concat Premises* (Eq. 3) when $\{\alpha_i\} = \mathbf{1}$.

Final Layer (FL): The final layer in the entailment stack usually outputs a single vector $\bar{h}$ which is then used in a linear layer and softmax to produce label probabilities. The join operation here is a weighted sum of the premise-level vectors. So we have $\tilde{X}_i = \bar{h}_i \quad \forall i$ and $\tilde{Y} = \sum_i \alpha_i \bar{h}_i$.

This is similar to a standard attention mechanism, where attended representation is computed by summing the scaled representations. However, such scaled addition is not possible when the outputs from lower layers are not of the same shapes, as in the following case.

Cross Attention Layer (CA): Cross-attention is a standard component of many entailment and reading comprehension models. This layer produces three outputs: (i) For each premise P_i, we get a hypothesis to premise cross attention matrix M^{hp_i} with shape (h $\times$ p_i), where h is the number of hypothesis tokens, and p_i is the number of tokens in premise P_i; (ii) for each premise P_i, we get a sequence of vectors $[\bar{P}_i]$ that corresponds to the token sequence of the premise P_i; and (iii) for the hypothesis, we get a single sequence of vectors $[\bar{H}_{qa}]$ that corresponds to its token sequence. M^{hp_i} attention matrix was generated by cross attention from $[\bar{H}_{qa}]$ to $[\bar{P}_i]$.

The join operation in this layer produces a cross attention matrix that spans the entire passage, i.e., has shape $(h \times p)$, where p is the total number of tokens across all premises. The operation first scales the cross-attention matrices by the sentence-relevance weights $\{\alpha_i\}$ in order to "tone down" the influence of distracting/irrelevant sentences, and then re-normalizes the final matrix:

$$\begin{aligned}\tilde{X}_i &= (M^{hp_i}, [\bar{P}_i], [\bar{H}_{qa}]) \quad \forall i\\ M^{hp} &= \left[\alpha_i M^{hp_1}; \quad \ldots; \quad \alpha_i M^{hp_n}\right]\\ M^{hp}_{ij} &= \frac{M^{hp}_{ij}}{\sum_k M^{hp}_{ik}}\\ [\bar{P}] &= \left[[\bar{P}_1]; \quad [\bar{P}_2]; \ldots; [\bar{P}_n]\right]\\ \tilde{Y} &= (M^{hp}, \bar{P}, \bar{H}_{qa})\end{aligned}$$

where M^{hp}_{ij} is i^{th} row and j^{th} column of M^{hp}.

Multee's multi-layer aggregator module uses join operations at two levels: **Cross Attention Layer (CA)** and **Final Layer (FL)**. The two corresponding aggregators share parameters up till the lower of the two join layers (CA in this case), where they both operate at the sentence level. Above this layer, one aggregator switches to operating at the paragraph level, where it has its own, unshared parameters. In general, if Multee were to aggregate at layers $\ell_{i1}, \ell_{i2}, \ldots, \ell_{ik}$, then the aggregators with joins at layers ℓ and ℓ' respectively could share parameters at layers $1, \ldots, \min\{\ell, \ell'\}$.

3.3 Implementation Details

Multee uses the ESIM stack as the entailment function pre-trained on SNLI and MultiNLI for both the relevance module and for the multi-layer aggregator module. It uses aggregation at two-levels, one at the cross-attention level (CA) and one at the final layer (FL). All uses of the entailment function in Multee are initialized with the same pre-trained entailment model weights. The embedding layer and the BiLSTM layer process paragraph-level contexts but processing at higher layers are done either at premise level or paragraph-level depending on where the join operation is performed.

4 Experiments

Datasets: We evaluate Multee on two datasets, OpenBookQA (Mihaylov et al., 2018) and MultiRC (Khashabi et al., 2018), both of which are specifically designed to test reasoning over multiple sentences. MultiRC is paragraph-based multiple-choice QA dataset derived from varying topics where the questions are answerable based on information from the paragraph. In MultiRC, each question can have more than one correct answer choice, and so it can be viewed as a binary classification task (one prediction per answer choice), with 4,848 / 4,583 examples in Dev/Test sets. OpenBookQA, on the other hand, has multiple-choice science questions with exactly one correct answer choice and no associated paragraph. As a result, this dataset requires the relevant facts to be retrieved from auxiliary resources including the open book of facts released with the paper and other sources such as WordNet (Miller, 1995) and ConceptNet (Speer and Havasi, 2012). It contains 500 questions in the Dev and Test sets.

		OpenBookQA		MultiRC	
		Accuracy		F1a | F1m | EM	
		Dev	Test	Dev	Test
Entailment Based Models with ELMo	Multee	**56.2**	54.8	69.6 | 73.0 | **22.8**	**70.4** | **73.8** | **24.5**
	Concatenate Premises	47.2	47.6	68.3 | 71.3 | 17.9	68.5 | 72.5 | 18.0
	Max of Local Decisions	47.8	45.2	66.5 | 69.3 | 16.3	66.70 | 70.4 | 19.4
Entailment Based Models with GloVe	Multee	54.6	**55.8**	68.3 | 71.7 | 16.4	69.9 | 73.6 | 19.0
	Concatenate Premises	47.4	42.6	66.9 | 70.7 | 14.8	68.7 | 72.4 | 16.3
	Max of Local Decisions	44.4	47.6	66.8 | 70.3 | 16.9	67.7 | 71.1 | 18.2
Previously Published Results	LR (Khashabi et al., 2018)	—	—	63.7 | 66.5 | 11.8	63.5 | 66.9 | 12.8
	IR (Khashabi et al., 2018)	—	—	60.0 | 64.3 |	54.8 | 53.9 |
	QM + ELMo (Mihaylov et al., 2018)	54.6	50.2	—	—
	ESIM + ELMo (Mihaylov et al., 2018)	53.9	48.9	—	—
	KER (Mihaylov et al., 2018)	55.6	51.4	—	—
Large Transformer Models	OFT (Sun et al., 2019)	—	52.0	67.2 | 69.3 | 15.2	—
	OFT (ensemble) (Sun et al., 2019)	—	52.8*	67.7 | 70.3 | 16.5*	—
	RS (Sun et al., 2019)	—	55.2*	69.2 | 71.5 | 22.6*	—
	RS (ensemble) (Sun et al., 2019)	—	**55.8***	**70.5** | **73.1** | 21.8*	—

Table 1: Comparison of Multee with other systems. Starred (*) results are based on contemporaneous system. Results marked (—) are not available. RS is Reading Strategies, KER is Knowledge Enhanced Reader, OFT is OpenAI FineTuned Transformer.

Preprocessing: For each question and answer choice, we create an answer hypothesis statement using a modified version of the script used in SciTail (Khot et al., 2018) construction. We wrote a handful of rules to better convert the question and answer to a hypothesis. We also mark the span of answer in the hypothesis with special begin and end tokens, `@@@answer` and `answer@@@` respectively[3]. For MultiRC, we also apply an off-the-shelf coreference resolution model[4] and replace the mentions when they resolve to pronouns occurring in a different sentence[5]. For OpenBookQA, we use the exact same retrieval as released by the authors of OpenBookQA[6] and use the OpenBook and WordNet as the knowledge source with top 5 sentences retrieved per query.

Training Multee: For OpenBookQA we use cross entropy loss for labels corresponding to 4 answer choices. For MultiRC, we use binary cross entropy loss for each answer-choice separately since in MultiRC each question can have more than one correct answer choice. The entailment components are pre-trained on sentence-level entailment tasks and then fine-tuned as part of end-to-end QA training. The MultiRC dataset includes sentence-level relevance labels. We supervise the Sentence Relevance module with a binary cross entropy loss for predicting these relevance labels when available. We used PyTorch (Paszke et al., 2017) and AllenNLP to implement our models and ran them on Beaker[7]. For pre-training we use the same hyper-parameters of ESIM(Chen et al., 2017) as available in implementation of AllenNLP (Gardner et al., 2017) and fine-tune the model parameters. We do not perform any hyper-parameter tuning for any of our models. We fine-tune all layers in ESIM except for the embedding layer.

Models Compared: We experiment with Glove (Pennington et al., 2014) and ELMo (Peters et al., 2018) embeddings for Multee and compare with following three types of systems:

(A) Baselines using entailment as a black-box We use the pre-trained entailment model as a black-box in two ways: concatenate premises (*Concat*) and aggregate sentence level decisions with a max operation (*Max*). Both models were also pre-trained on SNLI and MultiNLI datasets and fine-tuned on the target QA datasets with same

[3]Answer span marking gave substantial gains for all entailment based models including the baselines.

[4]https://github.com/huggingface/neuralcoref

[5]It is hard to learn co-reference, as these target datasets are too small to learn this in an end-to-end fashion.

[6]https://github.com/allenai/OpenBookQA

[7]`https://beaker.org/`

pre-processing.

(B) Previously published results: For MultiRC, there are two published baselines: IR (Information Retrieval) and LR (Logistic Regression). These simple models turn out to be strong baselines on this relatively smaller sized dataset. For OpenBookQA, we report published baselines from (Mihaylov et al., 2018): Question Match with ELMo (QM + ELMo), Question to Answer ESIM with ELMo (ESIM + ELMo) and their best result with the Knowledge Enhanced Reader (KER).

(C) Large Transformer based models: We compare with OpenAI-Transformer (OFT), pre-trained on large-scale language modeling task and fine-tuned on respective datasets. A contemporaneous work,[8] which published these transformer results, also fine-tuned this transformer further on a large scale reading comprehension dataset, RACE (Lai et al., 2017), before fine-tuning on the target QA datasets with their method, *Reading Strategies*.

4.1 Results

Table 1 summarizes the performance of all models. Multee outperforms the black-box entailment baselines (Concat and Max) that were pre-trained on the same data, previously published baselines, OpenAI transformer models. We note that the 95% confidence intervals around baseline accuracy for OpenBookQA and MultiRC are 4.3% and 1.3%, respectively.

On OpenBookQA test set, Multee with GloVe outperforms ensemble version of OpenAI transformer by 3.0 points in accuracy. It also outperforms single model version of Reading Strategies system and is comparable to their ensemble version. On MultiRC dev set, Multee with ELMo outperforms ensemble version of OpenAI transformer by 1.9 points in F1a, 2.7 in F1m and 6.3 in EM. It also outperforms single model version of Reading Strategies system and is comparable to their ensemble version. Recall that the Reading Strategies results are reported with an additional fine-tuning on another larger QA dataset, RACE (Lai et al., 2017) aside from the target QA datasets we use here.

While ELMo contextual embeddings helped in MultiRC, it did not help OpenBookQA. We believe this is in part due to the mismatch between our ELMo training setup where all sentences are treated as a single sequence, which, while true in MultiRC, is not the case in OpenBookQA.

[8] Published on arXiv on Oct 31, 2018 (Sun et al., 2019).

	OpenBookQA	MultiRC
	Accuracy	F1a | F1m
✗α_i	50.6	67.3 | 70.3
✓α_i	**55.8**	67.4 | 71.0
✓α_i + supervise	—	**68.3** | **71.7**

Table 2: Relevance Model Ablation of Multee. ✗α_i: without relevance weights, ✓α_i: with relevance weights respectively, ✓α_i + supervise: with supervised relevance weights. Test results on OpenBookQA and Dev results on MultiRC.

Aggregator	OpenBookQA	MultiRC
	Accuracy	F1a | F1m
Cross Attention (CA)	45.8	67.2 | 71.1
Final Layer (FL)	51.0	68.3 | 71.5
CA +FL	**55.8**	**68.3** | **71.7**

Table 3: Aggregator Level Ablation of Multee. On MultiRC, Multee uses relevance supervision but not on OpenBookQA because of unavailibility. Test results on OpenBookQA and Dev results on MultiRC.

In general, gains from Multee are more prominent in OpenBookQA than in MultiRC. We hypothesize that a key contributor to this difference is distraction being a lesser challenge in MultiRC, where premise sentences come from a single paragraph whose other sentences are often irrelevant and rarely distract towards incorrect answers. OpenBookQA has a noisier set of sentences, since an equal number of sentences is retrieved for the correct and each incorrect answer choice.

4.2 Ablations

Relevance Model Ablation. Table 2 shows the utility of the relevance module. We use the same setting as the full model (aggregation at Cross Attention (CA) and the Final Layer (FL)). As shown in the table, using the relevance module weights (✓α_i) leads to improved accuracy on both datasets (substantially so in OpenBookQA) as compared to ignoring the module, i.e., setting all weights to 1 (✗α_i). In MultiRC, we show that the additional supervision for the relevance module leads to even further improvements in score.

Multi-Level Aggregator Ablation. Multee performs aggregation at two levels: Cross Attention Layer (CA) and Final Layer (FL). We denote this by CA+FL. To show that multi-level aggregation is better than individual aggregations, we train

	OpenBookQA Accuracy	MultiRC F1a | F1m
Snli + MultiNli	**55.8**	**69.9** | **73.6**
Snli	50.4	69.3 | 73.3
Scratch	42.2	68.3 | 72.6

Table 4: Effect (on test data) of pre-training the entailment model used in Multee.

models with aggregation at only FL and at only CA. Table 3 shows that multi-layer aggregation is better than CA or FL alone on both the datasets.

4.3 Effect of Pre-training

One of the benefits of using entailment based components in a QA model is that we can pre-train them on large scale entailment datasets and fine-tune them as part of the QA model. Table 4 shows that such pre-training is valuable. The model trained from scratch is substantially worse in the case of OpenBookQA, highlighting the benefits of our entailment-based QA model.

Multee benefits come from two sources: (i) Re-purposing of entailment function for multi-sentence question answering, and (ii) transferring from a large-scale entailment task. In the case of OpenBookQA, both are helpful. For MultiRC, only the first is a significant contributor. Table 5 shows that re-purposing was a bigger factor for MultiRC, since Max and Concat models do not work well when trained from scratch.

		OpenBookQa Accuracy	MultiRc F1a | F1m
Max	Snli + MultiNli	**47.6**	**66.8** | **70.3**
	Scratch	32.4	42.8 | 44.0
Concat	Snli + MultiNli	**42.6**	**66.9** | **70.7**
	Scratch	35.8	51.3 | 50.4

Table 5: Pre-training ablations of black-box entailment baselines for OpenBookQA (test) and MultiRC (dev).

5 Analysis

Relevance Loss. The sentence-level relevance model provides a way to dig deeper into the overall QA model's behavior. When sentence-level supervision is available, as in the case of MultiRC, we can analyze the impact of different auxiliary losses for the relevance module. Table 6 shows the QA performance with different relevance losses, and Figure 5 shows a visualization of attention scores for a question in MultiRC. Overall, we find that two types of behaviors emerge from different loss functions. For instance, trying to minimize the sum of attention probability mass on irrelevant sentences i.e. $\sum_i \alpha_i(1 - y_i)$, called *IR Sum Loss*, causes the attention scores to become "peaky" i.e, high for one or two sentences, and close to zero for others. This leads to higher precision but at significantly lower recall for the QA system, as it now uses information from fewer but highly relevant sentences. Binary cross entropy loss (BCE) allows the model to attend to more relevant sentences thereby increasing recall without too much drop in precision.

	F1a precision	F1a recall
IR Sum Loss	**59.5**	68.5
BCE Loss	58.0	**83.2**

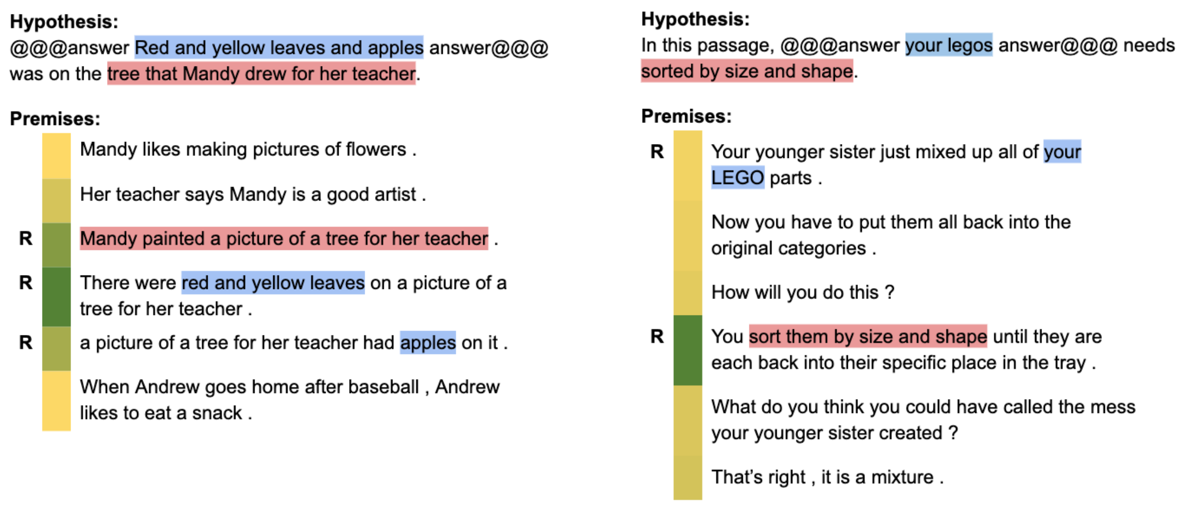
Table 6: F1a precision and recall on MultiRC Dev with 2 kinds of relevance losses. IR Sum is the sum of attention probability mass on irrelevant sentences. BCE is Binary Cross Entropy loss.

Failure Cases. As Figure 5 shows, our model with BCE loss tends to distribute the attention, especially to sentences close to the relevant ones. We hypothesize that the model is learning to use the contextualized BiLSTM representations to incorporate information from neighboring sentences, which is useful for this task and for passage understanding in general. For example, more than 60% of Dev questions in MultiRC have at least one adjacent relevant sentence pair. Figure 4a illustrates this behavior.

On the other hand, if the relevant sentences are far apart, the model finds it difficult to handle such long-range cross sentence dependencies in its contextualized representations. As a result, it ends up focusing attention on the most relevant sentence, missing out on other relevant sentences (Figure 4b). When these unattended but relevant sentences contain the answer, the model fails.

6 Related Work

Entailment systems have been applied to question-answering before but have only had limited success (Harabagiu and Hickl, 2006; Sacaleanu et al., 2008; Clark et al., 2012) in part because of the small size of the early entailment datasets (Dagan et al., 2006, 2013). Recent large scale entailment datasets such as SNLI (Bowman et al.,


Hypothesis:
@@@answer Red and yellow leaves and apples answer@@@ was on the tree that Mandy drew for her teacher.

Premises:

	Mandy likes making pictures of flowers .
	Her teacher says Mandy is a good artist .
R	Mandy painted a picture of a tree for her teacher .
R	There were red and yellow leaves on a picture of a tree for her teacher .
R	a picture of a tree for her teacher had apples on it .
	When Andrew goes home after baseball , Andrew likes to eat a snack .

Hypothesis:
In this passage, @@@answer your legos answer@@@ needs sorted by size and shape.

Premises:

R	Your younger sister just mixed up all of your LEGO parts .
	Now you have to put them all back into the original categories .
	How will you do this ?
R	You sort them by size and shape until they are each back into their specific place in the tray .
	What do you think you could have called the mess your younger sister created ?
	That's right , it is a mixture .

(a) Positive Example (b) Negative Example

Figure 4: Success and failure examples of Multee from MultiRC. **R**: annotated relevant sentences. Green/yellow: high/low predicted relevance.

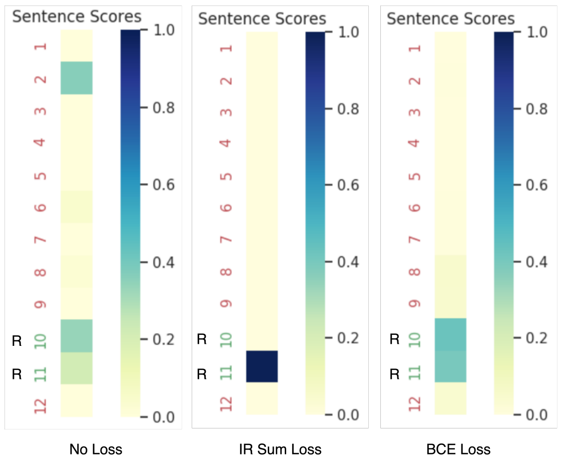


Figure 5: Sentence level attentions for various sentence relevance losses. **R**: annotated relevant sentences.

2015) and MultiNLI (Williams et al., 2018) have led to many new powerful neural entailment models that are not only more effective, but also produce better representations of sentences (Conneau et al., 2017). Models such as Decomposable Attention (Parikh et al., 2016) and ESIM (Chen et al., 2017), on the other hand, find alignments between the hypothesis and premise words through cross-attention. However, these improvements in entailment models have not yet translated to improvements in end tasks such as question answering.

SciTail (Khot et al., 2018) was created from a science QA task to push for models with a direct impact on QA. Entailment models trained on this dataset show minor improvements on the Aristo Reasoning Challenge (Clark et al., 2018; Musa et al., 2018). However, these QA systems make independent predictions and can not combine information from multiple supporting sentences.

Combining information from multiple sentences is a key problem in language understanding. Recent Reading comprehension datasets (Welbl et al., 2018; Khashabi et al., 2018; Yang et al., 2018; Mihaylov et al., 2018) explicitly evaluate a system's ability to perform such reasoning through questions that need information from multiple sentences in a passage. Most approaches on these tasks perform simple attention-based aggregation (Mihaylov et al., 2018; Song et al., 2018; Cao et al., 2018) and do not exploit the entailment models trained on large scale datasets.

7 Conclusions

Using entailment for question answering has seen limited success. Neural entailment models are designed and trained on tasks defined over sentence pairs, whereas QA often requires reasoning over longer texts spanning multiple sentences. We propose Multee, a novel QA model that addresses this mismatch. It uses an existing entailment model to both focus on relevant sentences and aggregate information from these sentences. Results on two challenging QA datasets, as well as our ablation study, indicate that entailment based QA can achieve state-of-the-art performance and is a promising direction for further research.

Acknowledgements

This work is supported in part by the National Science Foundation under Grant IIS-1815358. The computations on beaker.org were supported in part by credits from Google Cloud.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2018. Question answering by reasoning across documents with graph convolutional networks. *CoRR* abs/1808.09920.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1657–1668. https://doi.org/10.18653/v1/P17-1152.

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. *arXiv preprint arXiv:1803.05457* .

Peter Clark, Philip Harrison, and Xuchen Yao. 2012. An entailment-based approach to the qa4mre challenge. In *CLEF*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 670–680. https://www.aclweb.org/anthology/D17-1070.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, Springer, pages 177–190.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies* 6(4):1–220.

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.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 905–912.

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 *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of AAAI*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 785–794. https://doi.org/10.18653/v1/D17-1082.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2381–2391. http://aclweb.org/anthology/D18-1260.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM* 38(11):39–41.

Ryan Musa, Xiaoyan Wang, Achille Fokoue, Nicholas Mattei, Maria Chang, Pavan Kapanipathi, Bassem Makni, Kartik Talamadupula, and Michael Witbrock. 2018. Answering science exam questions using query rewriting with background knowledge. *CoRR* abs/1809.05726.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2249–2255. https://doi.org/10.18653/v1/D16-1244.

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 *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

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

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, Technical report, OpenAI.

Bogdan Sacaleanu, Constantin Orasan, Christian Spurk, Shiyan Ou, Oscar Ferrandez, Milen Kouylekov, and Matteo Negri. 2008. Entailment-based question answering for structured data. In *22nd International Conference on on Computational Linguistics: Demonstration Papers*. Association for Computational Linguistics, pages 173–176.

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.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA).

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. Improving machine reading comprehension with general reading strategies. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *IJCAI*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics* 6:287–302. http://aclweb.org/anthology/Q18-1021.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pages 1112–1122. https://doi.org/10.18653/v1/N18-1101.

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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.