Natural Language Comprehension with the EpiReader

Adam Trischler
adam.trischler

Zheng Ye
jeff.ye

Xingdi Yuan
eric.yuan

Philip Bachman
phil.bachman

Alessandro Sordoni
alessandro.sordoni

Kaheer Suleman
k.suleman

@maluuba.com
Maluuba Research
Montréal, Québec, Canada

Abstract

We present EpiReader, a novel model for machine comprehension of text. Machine comprehension of unstructured, real-world text is a major research goal for natural language processing. Current tests of machine comprehension pose questions whose answers can be inferred from some supporting text, and evaluate a model's response to the questions. EpiReader is an end-to-end neural model comprising two components: the first component proposes a small set of candidate answers after comparing a question to its supporting text, and the second component formulates hypotheses using the proposed candidates and the question, then reranks the hypotheses based on their estimated concordance with the supporting text. We present experiments demonstrating that EpiReader sets a new state-of-the-art on the CNN and Children's Book Test benchmarks, outperforming previous neural models by a significant margin.

1 Introduction

When humans reason about the world, we tend to formulate a variety of hypotheses and counterfactuals, then test them in turn by physical or thought experiments. The philosopher Epicurus first formalized this idea in his Principle of Multiple Explanations: if several theories are consistent with the observed data, retain them all until more data is observed. In this paper, we argue that the same principle can be applied to machine comprehension of natural language. We propose a deep neural comprehension model, trained end-to-end, that we call EpiReader.

Comprehension of natural language by machines, at a near-human level, is a prerequisite for an extremely broad class of useful applications of artificial intelligence. Indeed, most human knowledge is collected in the natural language of text. Machine comprehension (MC) has therefore garnered significant attention from the machine learning research community. Machine comprehension is typically evaluated by posing a set of questions based on a supporting text passage, then scoring a system's answers to those questions. Such tests are objectively gradable and may assess a range of abilities, from basic understanding to causal reasoning to inference (Richardson et al., 2013).

In the past year, two large-scale MC datasets have been released: the CNN/Daily Mail corpus, consisting of news articles from those outlets (Hermann et al., 2015), and the Children's Book Test (CBT), consisting of short excerpts from books available through Project Gutenberg (Hill et al., 2016). The size of these datasets (on the order of 10^5 distinct questions) makes them amenable to data-intensive deep learning techniques. Both corpora use Cloze-style questions (Taylor, 1953), which are formulated by replacing a word or phrase in a given sentence with a placeholder token. The task is then to find the answer that "fills in the blank".

In tandem with these corpora, a host of neural machine comprehension models has been developed (Weston et al., 2015b; Hermann et al., 2015; Hill et al., 2016; Kadlec et al., 2016; Chen et al., 2016). We compare EpiReader to these earlier models through training and evaluation on the CNN and

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 128–137,
Austin, Texas, November 1-5, 2016. ©2016 Association for Computational Linguistics

CBT datasets.[1]

EpiReader factors into two components. The first component extracts a small set of potential answers based on a shallow comparison of the question with its supporting text; we call this the *Extractor*. The second component reranks the proposed answers based on deeper semantic comparisons with the text; we call this the *Reasoner*. We can summarize this process as *Extract* → *Hypothesize* → *Test*[2]. The semantic comparisons implemented by the Reasoner are based on the concept of *recognizing textual entailment* (RTE) (Dagan et al., 2006), also known as natural language inference. This process is computationally demanding. Thus, the Extractor serves the important function of filtering a large set of potential answers down to a small, tractable set of likely candidates for more thorough testing. The two-stage process is an analogue of *structured prediction cascades* (Weiss and Taskar, 2010), wherein a sequence of increasingly complex models progressively filters the output space in order to trade off between model complexity and limited computational resources. We demonstrate that this cascade-like framework is applicable to machine comprehension and can be trained end-to-end with stochastic gradient descent.

The Extractor follows the form of a pointer network (Vinyals et al., 2015), and uses a differentiable attention mechanism to indicate words in the text that potentially answer the question. This approach was used (on its own) for question answering with the Attention Sum Reader (Kadlec et al., 2016). The Extractor outputs a small set of answer candidates along with their estimated probabilities of correctness. The Reasoner forms hypotheses by inserting the candidate answers into the question, then estimates the concordance of each hypothesis with each sentence in the supporting text. We use these estimates as a measure of the evidence for a hypothesis, and aggregate evidence over all sentences. In the end, we combine the Reasoner's evidence with the Extractor's probability estimates to produce a final ranking of the answer candidates.

This paper is organized as follows. In Section 2 we formally define the problem to be solved and give some background on the datasets used in our tests. In Section 3 we describe EpiReader, focusing on its two components and how they combine. Section 4 discusses related work, and Section 5 details our experimental results and analysis. We conclude in Section 6.

2 Problem definition, notation, datasets

EpiReader's task is to answer a Cloze-style question by reading and comprehending a supporting passage of text. The training and evaluation data consist of tuples $(\mathcal{Q}, \mathcal{T}, a^*, A)$, where $\mathcal{Q}$ is the question (a sequence of words $\{q_1, ...q_{|\mathcal{Q}|}\}$), $\mathcal{T}$ is the text (a sequence of words $\{t_1, ..., t_{|\mathcal{T}|}\}$), A is a set of possible answers $\{a_1, ..., a_{|A|}\}$, and $a^* \in A$ is the correct answer. All words come from a vocabulary V, and $A \subset \mathcal{T}$. In each question, there is a placeholder token indicating the missing word to be filled in.

2.1 Datasets

CNN This corpus is built using articles scraped from the CNN website. The articles themselves form the text passages, and questions are generated synthetically from short summary statements that accompany each article. These summary points are (presumably) written by human authors. Each question is created by replacing a named entity in a summary point with a placeholder token. All named entities in the articles and questions are replaced with anonymized tokens that are shuffled for each $(\mathcal{Q}, \mathcal{T})$ pair. This forces the model to rely only on the text, rather than learning world knowledge about the entities during training. The CNN corpus (henceforth CNN) was presented by Hermann et al. (2015).

Children's Book Test This corpus is constructed similarly to CNN, but from children's books available through Project Gutenberg. Rather than articles, the text passages come from book excerpts of 20 sentences. Since no summaries are provided, a question is generated by replacing a single word in the next (i.e. 21st) sentence. The corpus distinguishes questions based on the type of word that is replaced: named entity, common noun, verb, or preposition. Like Kadlec et al. (2016), we focus only on the first two classes since Hill et al. (2016) showed that stan-

[1] The CNN and Daily Mail datasets were released together and have the same form. The Daily Mail dataset is significantly larger; therefore, models consistently score higher when trained/tested on it.

[2] The Extractor performs extraction, while the Reasoner both hypothesizes and tests.

dard LSTM language models already achieve human-level performance on the latter two. Unlike in the CNN corpora, named entities are not anonymized and shuffled in the Children's Book Test (CBT). CBT was presented by Hill et al. (2016).

The different methods of construction for questions in each corpus mean that CNN and CBT assess different aspects of comprehension. The summary points of CNN are a condensed paraphrasing of information in the text; thus, determining the correct answer relies mostly on recognizing textual entailment. On the other hand, CBT is about story prediction. It is a comprehension task insofar as comprehension is likely necessary for story prediction, but comprehension alone may not be sufficient. Indeed, there are some CBT questions that are unanswerable given the preceding context.

3 EpiReader

3.1 Overview and intuition

EpiReader explicitly leverages the observation that the answer to a question is often a word or phrase from the related text passage. This condition holds for the CNN and CBT datasets. EpiReader's first module, the Extractor, can thus select a small set of candidate answers by pointing to their locations in the supporting passage. This mechanism is detailed in Section 3.2, and was used previously by the Attention Sum Reader (Kadlec et al., 2016). Pointing to candidate answers removes the need to apply a softmax over the entire vocabulary as in Weston et al. (2015b), which is computationally more costly and uses less-direct information about the context of a predicted answer in the supporting text.

EpiReader's second module, the Reasoner, begins by formulating hypotheses using the extracted answer candidates. It generates each hypothesis by replacing the placeholder token in the question with an answer candidate. Cloze-style questions are ideally-suited to this process, because inserting the correct answer at the placeholder location produces a well-formed, grammatical statement. Thus, the correct hypothesis will "make sense" to a language model.

The Reasoner then tests each hypothesis individually. It compares a hypothesis to the text, split into sentences, to measure textual entailment, and then aggregates entailment over all sentences. This computation uses a pair of convolutional encoder networks followed by a recurrent neural network. The convolutional encoders generate abstract representations of the hypothesis and each text sentence; the recurrent network estimates and aggregates entailment. This is described formally in Section 3.3. The end-to-end EpiReader model, combining the Extractor and Reasoner modules, is depicted in Figure 1.

Throughout our model, words will be represented with trainable embeddings (Bengio et al., 2000). We represent these embeddings using a matrix $\mathbf{W} \in \mathbb{R}^{D\times|V|}$, where D is the embedding dimension and $|V|$ is the vocabulary size.

3.2 The Extractor

The Extractor is a Pointer Network (Vinyals et al., 2015). It uses a pair of bidirectional recurrent neural networks, $f(\theta_T, \mathbf{T})$ and $g(\theta_Q, \mathbf{Q})$, to encode the text passage and the question. θ_T represents the parameters of the text encoder, and $\mathbf{T} \in \mathbb{R}^{D\times N}$ is a matrix representation of the text (comprising N words), whose columns are individual word embeddings $\mathbf{t}_i$. Likewise, θ_Q represents the parameters of the question encoder, and $\mathbf{Q} \in \mathbb{R}^{D\times N_Q}$ is a matrix representation of the question (comprising N_Q words), whose columns are individual word embeddings $\mathbf{q}_j$.

We use a recurrent neural network with gated recurrent units (GRU) (Bahdanau et al., 2015) to scan over the columns (i.e. word embeddings) of the input matrix. We selected the GRU because it is computationally simpler than Long Short-Term Memory (Hochreiter and Schmidhuber, 1997), while still avoiding the problem of vanishing/exploding gradients often encountered when training recurrent networks.

The GRU's hidden state gives a representation of the ith word conditioned on preceding words. To include context from proceeding words, we run a second GRU over $\mathbf{T}$ in the reverse direction. We refer to the combination as a biGRU. At each step the biGRU outputs two d-dimensional encoding vectors, one for the forward direction and one for the backward direction. We concatenate these to yield a vector $f(\mathbf{t}_i) \in \mathbb{R}^{2d}$. The question biGRU is similar, but we form a single-vector representation of the question by concatenating the final forward state with the final backward state, which we denote $g(\mathbf{Q}) \in \mathbb{R}^{2d}$.

As in Kadlec et al. (2016), we model the probability that the ith word in text $\mathcal{T}$ answers question $\mathcal{Q}$

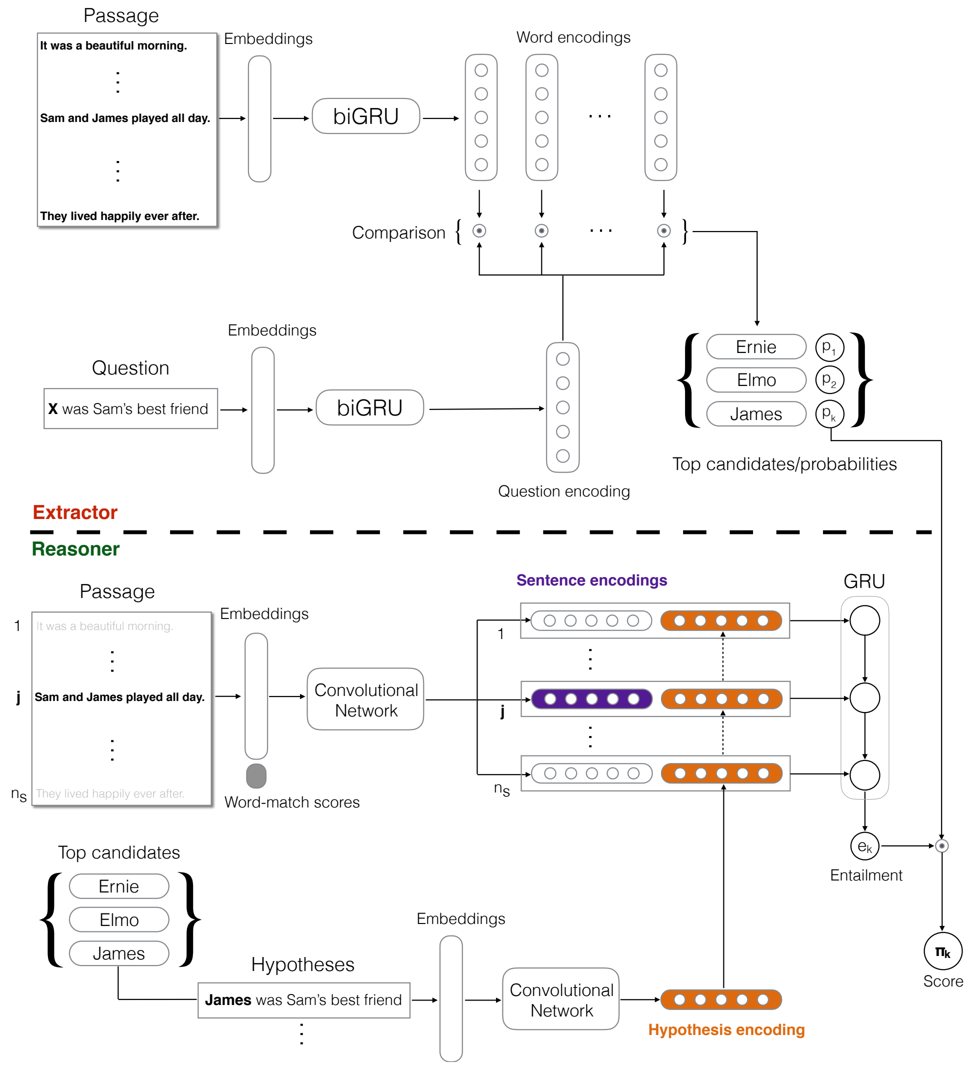


Figure 1: The complete EpiReader framework. The Extractor is above, the Reasoner below. Propagating the Extractor's probability estimates forward and combining them with the Reasoner's entailment estimates renders the model end-to-end differentiable.

using

$$s_i \propto \exp(f(\mathbf{t}_i) \cdot g(\mathbf{Q})), \tag{1}$$

which takes the inner product of the text and question representations followed by a softmax. In many cases unique words repeat in a text. Therefore, we compute the total probability that word w is the correct answer using a sum:

$$P(w \,|\, \mathcal{T}, \mathcal{Q}) = \sum_{i:\ t_i = w} s_i. \tag{2}$$

This probability is evaluated for each unique word in $\mathcal{T}$. Finally, the Extractor outputs the set $\{p_1, ..., p_K\}$ of the K highest word probabilities from 2, along with the corresponding set of K most probable answer words $\{\hat{a}_1, ..., \hat{a}_K\}$.

3.3 The Reasoner

The indicial selection involved in gathering $\{\hat{a}_1, ..., \hat{a}_K\}$, which is equivalent to a K-best $\arg\max$, is not a continuous function of its inputs. To construct an end-to-end differentiable model, we bypass this by propagating the probability estimates of the Extractor directly through the Reasoner.

The Reasoner begins by inserting the answer candidates, which are single words or phrases, into the question sequence $\mathcal{Q}$ at the placeholder location. This forms K hypotheses $\{\mathcal{H}_1, ..., \mathcal{H}_K\}$. At this point, we consider each hypothesis to have probability $p(\mathcal{H}_k) \approx p_k$, as estimated by the Extractor. The Reasoner updates and refines this estimate.

The hypotheses represent new information in some sense—they are statements we have constructed, albeit from words already present in the question and text passage. The Reasoner estimates entailment between the statements $\mathcal{H}_k$ and the passage $\mathcal{T}$. We denote these estimates using $e_k = F(\mathcal{H}_k, \mathcal{T})$, with F to be defined. We start by reorganizing $\mathcal{T}$ into a sequence of N_s sentences: $\mathcal{T} = \{t_1, \ldots, t_N\} \rightarrow \{\mathcal{S}_1, \ldots, \mathcal{S}_{N_s}\}$, where $\mathcal{S}_i$ is a sequence of words.

For each hypothesis and each sentence of the text, Reasoner input consists of two matrices: $\mathbf{S}_i \in \mathbb{R}^{D\times|\mathcal{S}_i|}$, whose columns are the embedding vectors for each word of sentence $\mathcal{S}_i$, and $\mathbf{H}_k \in \mathbb{R}^{D\times|\mathcal{H}_k|}$, whose columns are the embedding vectors for each word in the hypothesis $\mathcal{H}_k$. The embedding vectors themselves come from matrix $\mathbf{W}$, as before.

These matrices feed into a convolutional architecture based on that of Severyn and Moschitti (2016). The architecture first augments $\mathbf{S}_i$ with matrix $\mathbf{M} \in \mathbb{R}^{2\times|\mathcal{S}_i|}$. The first row of $\mathbf{M}$ contains the inner product of each word embedding in the sentence with the candidate answer embedding, and the second row contains the maximum inner product of each sentence word embedding with any word embedding in the question. These word-matching features were inspired by similar approaches in Wang and Jiang (2016) and Trischler et al. (2016), where they were shown to improve entailment estimates.

The augmented $\mathbf{S}_i$ is then convolved with a bank of filters $\mathbf{F}^S \in \mathbb{R}^{(D+2)\times m}$, while $\mathbf{H}_k$ is convolved with filters $\mathbf{F}^H \in \mathbb{R}^{D\times m}$, where m is the convolutional filter width. We add a bias term and apply a nonlinearity (we use a ReLU) following the convolution. Maxpooling over the sequences then yields two vectors: the representation of the text sentence, $\mathbf{r}_{\mathcal{S}_i} \in \mathbb{R}^{N_F}$, and the representation of the hypothesis, $\mathbf{r}_{\mathcal{H}_k} \in \mathbb{R}^{N_F}$, where N_F is the number of filters.

We then compute a scalar similarity score between these vector representations using the bilinear form

$$\varsigma = \mathbf{r}_{\mathcal{S}_i}^T \mathbf{R} \mathbf{r}_{\mathcal{H}_k}, \qquad (3)$$

where $\mathbf{R} \in \mathbb{R}^{N_F\times N_F}$ is a matrix of trainable parameters. We then concatenate the similarity score with the sentence and hypothesis representations to get a vector, $\mathbf{x}_{ik} = [\varsigma; \mathbf{r}_{\mathcal{S}_i}; \mathbf{r}_{\mathcal{H}_k}]^T$. There are more powerful models of textual entailment that could have been used in place of this convolutional architecture. We adopted the approach of Severyn and Moschitti (2016) for computational efficiency.

The resulting sequence of N_s vectors feeds into yet another GRU for synthesis, of hidden dimension d_S. Intuitively, it is often the case that evidence for a particular hypothesis is distributed over several sentences. For instance, if we hypothesize that *the football is in the park*, perhaps it is because one sentence tells us that *Sam picked up the football* and a later one tells us that *Sam ran to the park*.[3] The Reasoner synthesizes distributed information by running a GRU network over $\mathbf{x}_{ik}$, where i indexes sentences and represents the step dimension.[4] The final hidden state of the GRU is fed through a fully-connected layer, yielding a single scalar y_k. This value represents the collected evidence for $\mathcal{H}_k$ based on the text. In practice, the Reasoner processes all K hypotheses in parallel and the estimated entailment of each is normalized by a softmax, $e_k \propto \exp(y_k)$.

As pointed out in Kadlec et al. (2016), it is a strength of the pointer framework that it does not blend the representations that are being attended. Contrast this with typical attention mechanisms where such a blended representation is used downstream to make similarity comparisons with, *e.g.*, output vectors.

Differentiable attention mechanisms (as in Bahdanau et al. (2015), for example) typically blend internal representations together through a weighted sum, then use this 'blend' downstream for similarity comparisons. The pointer framework does not resort to this blending; Kadlec et al. (2016) explain that this is an advantage, since in comprehension tasks the goal is to select the correct answer among semantically similar candidates and more exact matching is necessary. The reranking function performed by the Reasoner entails this advantage, by examining the separate hypotheses individually without blending.

[3] This example is characteristic of the *bAbI* dataset (Weston et al., 2015a).

[4] Note a benefit of forming the hypothesis: it renders bidirectional aggregation unnecessary, since knowing both the question and the putative answer "closes the loop" the same way that a bidirectional encoding would.

3.4 Combining components

Finally, we combine the evidence from the Reasoner with the probability from the Extractor. We compute the output probability of each hypothesis, π_k, according to the product

$$\pi_k \propto e_k p_k, \tag{4}$$

whereby the evidence of the Reasoner can be interpreted as a correction to the Extractor probabilities, applied as an additive shift in log-space. We experimented with other combinations of the Extractor and Reasoner, but we found the multiplicative approach to yield the best performance.

After combining results from the Extractor and Reasoner to get the probabilities π_k described in Eq. 4, we optimize the parameters of the full EpiReader to minimize a cost comprising two terms, $\mathcal{L}_E$ and $\mathcal{L}_R$. The first term is a standard negative log-likelihood objective, which encourages the Extractor to rate the correct answer above other answers. This is the same loss term used in Kadlec et al. (2016). It is given by:

$$\mathcal{L}_E = \mathop{\mathbb{E}}_{(\mathcal{Q},\mathcal{T},a^*,A)} \left[-\log P(a^* \,|\, \mathcal{T}, \mathcal{Q}) \right], \tag{5}$$

where $P(a^* \,|\, \mathcal{T}, \mathcal{Q})$ is as defined in Eq. 2, and a^* denotes the true answer. The second term is a margin-based loss on the end-to-end probabilities π_k. We define π^* as the probability π_k corresponding to the true answer word a^*. This term is given by:

$$\mathcal{L}_R = \mathop{\mathbb{E}}_{(\mathcal{Q},\mathcal{T},a^*,A)} \left[\sum_{\hat{a}_i \in \{\hat{a}_1,\ldots,\hat{a}_K\} \backslash a^*} [\gamma - \pi^* + \pi_{\hat{a}_i}]_+ \right], \tag{6}$$

where γ is a margin hyperparameter, $\{\hat{a}_1, ..., \hat{a}_K\}$ is the set of K answers proposed by the Extractor, and $[x]_+$ indicates truncating x to be non-negative. Intuitively, this loss says that we want the end-to-end probability π^* for the correct answer to be at least γ larger than the probability $\pi_{\hat{a}_i}$ for any other answer proposed by the Extractor. During training, the correct answer is occasionally missed by the Extractor, especially in early epochs. We counter this issue by forcing the correct answer into the top K set while training. When evaluating the model on validation and test examples we rely fully on the top K answers proposed by the Extractor.

To get the final loss term $\mathcal{L}_{ER}$, minus ℓ_2 regularization terms on the model parameters, we take a weighted combination of $\mathcal{L}_E$ and $\mathcal{L}_R$:

$$\mathcal{L}_{ER} = \mathcal{L}_E + \lambda \mathcal{L}_R, \tag{7}$$

where λ is a hyperparameter for weighting the relative contribution of the Extractor and Reasoner losses. In practice, we found that λ should be fairly large (*e.g.*, $10 < \lambda < 100$). Empirically, we observed that the output probabilities from the Extractor often peak and saturate the first softmax; hence, the Extractor term can come to dominate the Reasoner term without the weight λ (we discuss the Extractor's propensity to overfit in Section 5).

4 Related Work

The Impatient and Attentive Reader models were proposed by Hermann et al. (2015). The Attentive Reader applies bidirectional recurrent encoders to the question and supporting text. It then uses the attention mechanism described in Bahdanau et al. (2015) to compute a fixed-length representation of the text based on a weighted sum of the text encoder's output, guided by comparing the question representation to each location in the text. Finally, a joint representation of the question and supporting text is formed by passing their separate representations through a feed-forward MLP and an answer is selected by comparing the MLP output to a representation of each possible answer. The Impatient Reader operates similarly, but computes attention over the text after processing each consecutive word of the question. The two models achieved similar performance on the CNN and Daily Mail datasets.

Memory Networks were first proposed by Weston et al. (2015b) and later applied to machine comprehension by Hill et al. (2016). This model builds fixed-length representations of the question and of windows of text surrounding each candidate answer, then uses a weighted-sum attention mechanism to combine the window representations. As in the previous Readers, the combined window representation is then compared with each possible answer to form a prediction about the best answer. What distinguishes Memory Networks is how they construct the question and text window representations. Rather than a recurrent network, they use a specially-designed, trainable transformation of the word embeddings.

Most of the details for the very recent AS Reader are provided in the description of our Extractor module in Section 3.2, so we do not summarize it further here. This model (Kadlec et al., 2016) set the previous state-of-the-art on the CBT dataset.

During the write-up of this paper, another very recent model came to our attention. Chen et al. (2016) propose using a bilinear term instead of a $\tanh$ layer to compute the attention between question and passage words, and also uses the attended word encodings for direct, pointer-style prediction as in Kadlec et al. (2016). This model set the previous state-of-the-art on the CNN dataset. However, this model used embedding vectors pretrained on a large external corpus (Pennington et al., 2014).

EpiReader borrows ideas from other models as well. The Reasoner's convolutional architecture is based on Severyn and Moschitti (2016) and Blunsom et al. (2014). Our use of word-level matching was inspired by the Parallel-Hierarchical model of Trischler et al. (2016) and the natural language inference model of Wang and Jiang (2016). Finally, the idea of formulating and testing hypotheses for question-answering was used to great effect in IBM's DeepQA system for *Jeopardy!* (Ferrucci et al., 2010) (although that was a more traditional information retrieval pipeline rather than an end-to-end neural model), and also resembles the framework of structured prediction cascades (Weiss and Taskar, 2010).

5 Evaluation

5.1 Implementation and training details

To train our model we used stochastic gradient descent with the ADAM optimizer (Kingma and Ba, 2015), with an initial learning rate of 0.001. The word embeddings were initialized randomly, drawing from the uniform distribution over $[-0.05, 0.05)$. We used batches of 32 examples, and early stopping with a patience of 2 epochs. Our model was implemented in Theano (Bergstra et al., 2010) using the Keras framework (Chollet, 2015).

The results presented below for EpiReader were obtained by searching over a small grid of hyperparameter settings. We selected the model that, on each dataset, maximized accuracy on the validation set, then evaluated it on the test set. We record the best settings for each dataset in Table 1. As has been done previously, we train separate models on CBT's named entity (CBT-NE) and common noun (CBT-CN) splits. All our models used ℓ_2-regularization at 0.001, $\lambda = 50$, and $\gamma = 0.04$. We did not use dropout but plan to investigate its effect in the future. Hill et al. (2016) and Kadlec et al. (2016) also present results for ensembles of their models. Time did not permit us to generate an ensemble of EpiReaders on the CNN dataset so we omit those measures; however, EpiReader ensembles (of seven models) demonstrated improved performance on the CBT dataset.

Table 1: Hyperparameter settings for best EpiReaders. D is the embedding dimension, d is the hidden dimension in the Extractor GRUs, K is the number of candidates to consider, m is the filter width, N_F is the number of filters, and d_S is the hidden dimension in the Reasoner GRU.

	Hyperparameters					
Dataset	D	d	K	m	N_F	d_S
CBT-NE	300	128	5	3	16	32
CBT-CN	300	128	5	3	32	32
CNN	384	256	10	3	32	32

5.2 Results

In Table 5.2, we compare the performance of EpiReader against that of several baselines, on the validation and test sets of the CBT and CNN corpora. We measure EpiReader performance at the output of both the Extractor and the Reasoner. EpiReader achieves state-of-the-art performance across the board for both datasets. On CNN, we score 2.2% higher on test than the best previous model of Chen et al. (2016). Interestingly, an analysis of the CNN dataset by Chen et al. (2016) suggests that approximately 25% of the test examples contain coreference errors or questions which are "ambiguous/hard" even for a human analyst. If this estimate is accurate, then EpiReader, achieving an absolute test accuracy of 74.0%, is operating close to expected human performance. On the other hand, ambiguity is unlikely to be distributed evenly over entities, so a good model should be able to perform at better-than-chance levels even on questions where the correct answer is uncertain. If, on the 25% of "noisy" questions, the model can shift its hit rate from, *e.g.*, 1/10 to 1/3, then there is still a fair amount of performance to gain.

	CBT-NE		CBT-CN	
Model	valid	test	valid	test
Humans (context + query) [1]	-	81.6	-	81.6
LSTMs (context + query) [1]	51.2	41.8	62.6	56.0
MemNNs [1]	70.4	66.6	64.2	63.0
AS Reader [2]	73.8	68.6	68.8	63.4
EpiReader Extractor	73.2	69.4	69.9	66.7
EpiReader	**75.3**	**69.7**	**71.5**	**67.4**
AS Reader (ensemble) [2]	74.5	70.6	71.1	68.9
EpiReader (ensemble)	**76.6**	**71.8**	**73.6**	**70.6**

	CNN	
Model	valid	test
Deep LSTM Reader [3]	55.0	57.0
Attentive Reader [3]	61.6	63.0
Impatient Reader [3]	61.8	63.8
MemNNs [1]	63.4	66.8
AS Reader [2]	68.6	69.5
Stanford AR [4]	72.4	72.4
EpiReader Extractor	71.8	72.0
EpiReader	**73.4**	**74.0**

Table 2: Model comparison on the CBT and CNN datasets. Results marked with [1] are from Hill et al. (2016), with [2] are from Kadlec et al. (2016), with [3] are from Hermann et al. (2015), and with [4] are from Chen et al. (2016).

Ablated component	Validation accuracy (%)
-	**71.5**
Word-match scores	70.3
Bilinear similarity	70.0
Reasoner	68.7
Convolutional encoders	71.0

Table 3: Ablation study on CBT-CN validation set.

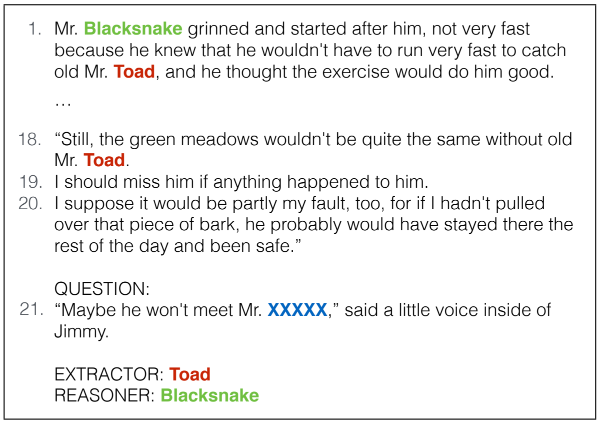
On CBT-CN our single model scores 4.0% higher than the previous best of the AS Reader. The improvement on CBT-NE is more modest at 1.1%. Looking more closely at our CBT-NE results, we found that the validation and test accuracies had relatively high variance even in late epochs of training. We discovered that many of the validation and test questions were asked about the same named entity, which may explain this issue.

5.3 Analysis

We measure the contribution of several components of the Reasoner by ablating them. Results on the validation set of CBT-CN are presented in Table 3. The word-match scores (cosine similarities stored in the first two rows of matrix $\mathbf{M}$, see Section 3.3) make a contribution of 1.2% to the validation performance, indicating that they are useful. Similarly, the bilinear similarity score ς, which is passed to the final GRU network, contributes 1.5%.

Removing the Reasoner altogether reduces our model to the AS Reader, whose results we have reproduced to within negligible difference. Aside from achieving state-of-the-art results at its final output, the EpiReader framework gives a boost to its Extractor component through the joint training process. This can be seen by referring back to Table 5.2, wherein we also provide accuracy scores evaluated at the output of the Extractor. These are all higher than the analogous scores reported for the AS Reader. Based on our own work with that model, we found it to overfit the training set rapidly and significantly, achieving training accuracy scores upwards of 98% after only 2 epochs. We suspect that the Reasoner module had a regularizing effect on the Extractor, but leave the confirmation for future work.

Although not exactly an ablation, we also tried bypassing the Reasoner's convolutional encoders altogether, along with the word-match scores and the bilinear similarity. This was done as follows: from the Extractor, we pass to the Reasoner's final GRU (i) the bidirectional hidden representation of the question; (ii) the bidirectional hidden representations of the *end* of each story sentence (recall that the Reasoner operates on sentence representations). Thus, we reuse (parts of) the original biGRU encodings. This cuts down on the number of model parameters and on the length of the graph through which gradients must flow, potentially providing a stronger learning signal to the initial encoders. We found that this change yielded a relatively small reduction in performance on CBT-CN, perhaps for the reasons just discussed—only 0.5%, as given in the final line of



1. Mr. Blacksnake grinned and started after him, not very fast because he knew that he wouldn't have to run very fast to catch old Mr. Toad, and he thought the exercise would do him good.

…

18. “Still, the green meadows wouldn't be quite the same without old Mr. Toad.
19. I should miss him if anything happened to him.
20. I suppose it would be partly my fault, too, for if I hadn't pulled over that piece of bark, he probably would have stayed there the rest of the day and been safe.”

QUESTION:
21. “Maybe he won't meet Mr. XXXXX,” said a little voice inside of Jimmy.

EXTRACTOR: Toad
REASONER: Blacksnake

Figure 2: An abridged example from CBT-NE demonstrating corrective reranking by the Reasoner.

Table 3. This suggests that competitive performance may be achieved with other, simpler architectures for the Reasoner's entailment system and this will be the subject of future research.

An analysis by Kadlec et al. (2016) indicates that the trained AS Reader includes the correct answer among its five most probable candidates on approximately 95% of test examples for both datasets. We verified that our Extractor achieved a similar rate, and of course this is vital for performance of the full system, since the Reasoner cannot recover when the correct answer is not among its inputs.

Our results show that the Reasoner often corrects erroneous answers from the Extractor. Figure 2 gives an example of this correction. In the text passage, from CBT-NE, Mr. Blacksnake is pursuing Mr. Toad, presumably to eat him. The dialogue in the question sentence refers to both: Mr. Toad is its subject, referred to by the pronoun “he”, and Mr. Blacksnake is its object. In the preceding sentences, it is clear (to a human) that Jimmy is worried about Mr. Toad and his potential encounter with Mr. Blacksnake. The Extractor, however, points most strongly to “Toad”, possibly because he has been referred to most recently. The Reasoner corrects this error and selects “Blacksnake” as the answer. This relies on a deeper understanding of the text. The named entity can, in this case, be inferred through an alternation of the entities most recently referred to. This kind alternation is typical of dialogues, when two actors interact in turns. The Reasoner can capture this behavior because it examines sentences in sequence.

6 Conclusion

We presented the novel EpiReader framework for machine comprehension and evaluated it on two large, complex datasets: CNN and CBT. Our model achieves state-of-the-art results on these corpora, outperforming all previous approaches. In future work, we plan to test our framework with alternative models for natural language inference (*e.g.*, Wang and Jiang (2016)), and explore the effect of pretraining such a model specifically on an inference task.

As a general framework that consists in a two-stage cascade, EpiReader can be implemented using a variety of mechanisms in the Extractor and Reasoner stages. We have demonstrated that this cascade-like framework is applicable to machine comprehension and can be trained end-to-end. As more powerful machine comprehension models inevitably emerge, it may be straightforward to boost their performance using EpiReader's structure.

References

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

[Bengio et al.2000] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, pages 932–938.

[Bergstra et al.2010] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *In Proc. of SciPy*.

[Blunsom et al.2014] Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences.

[Chen et al.2016] Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn / daily mail reading comprehension task. In *Association for Computational Linguistics (ACL).*

[Chollet2015] François Chollet. 2015. keras. `https://github.com/fchollet/keras`.

[Dagan et al.2006] 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 textual entailment*, pages 177–190. Springer.

[Ferrucci et al.2010] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

[Hermann et al.2015] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

[Hill et al.2016] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. *ICLR*.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Kadlec et al.2016] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*.

[Kingma and Ba2015] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. *Proc. EMNLP*, 12.

[Richardson et al.2013] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 1, page 2.

[Severyn and Moschitti2016] Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. *arXiv preprint arXiv:1604.01178*.

[Taylor1953] Wilson L Taylor. 1953. Cloze procedure: a new tool for measuring readability. *Journalism and Mass Communication Quarterly*, 30.

[Trischler et al.2016] Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Philip Bachman, and Kaheer Suleman. 2016. A parallel-hierarchical model for machine comprehension on sparse data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

[Vinyals et al.2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2674–2682.

[Wang and Jiang2016] Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. *NAACL*.

[Weiss and Taskar2010] David J Weiss and Benjamin Taskar. 2010. Structured prediction cascades. In *AISTATS*, pages 916–923.

[Weston et al.2015a] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015a. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

[Weston et al.2015b] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015b. Memory networks. *ICLR*.