Tell Me Why: Using Question Answering as Distant Supervision for Answer Justification

Anonymous EMNLP submission

Abstract

For many applications of question answering (QA), being able to explain why a given model chose an answer is critical. However, the lack of labeled data for answer justifications makes learning this difficult and expensive. Here we propose an approach that uses answer ranking as distant supervision for learning how to select informative justifications. We propose a neural network architecture for QA that reranks answer justifications as an intermediate (and human-interpretable) step in answer selection. Our approach is informed by a set of features designed to combine both learned representations and explicit features to capture the connection between questions, answers, and answer justifications. We show that with this end-to-end approach we are able to significantly improve upon a strong IR baseline in both justification ranking (+9% rated highly relevant) and answer selection (+6% P@1).

1 Introduction

Developing interpretable machine learning (ML) models, that is, models where a human user can *understand* what the model is learning, is considered by many to be crucial for ensuring usability and accelerating progress (Craven and Shavlik, 1996; Kim et al., 2015; Letham et al., 2015; Ribeiro et al., 2016). For many applications of question answering (QA), i.e., finding short answers to natural language questions, simply providing an answer is not sufficient. A complete approach must be interpretable, i.e., able to *explain* why an answer is correct. For example, in the medical domain, a QA approach that answers treatment questions would not be trusted if the treatment recommendation is not explained in terms that can be understood by the human user.

Question: Which of these is a response to an internal stimulus?
(A) A sunflower turns to face the rising sun.
(B) A cucumber tendril wraps around a wire.
(C) A pine tree knocked sideways in a landslide grows upward in a bend.
(D) Guard cells of a tomato plant leaf close when there is little water in the roots .

Justification: Plants rely on hormones to send signals within the plant in order to respond to internal stimuli such as a lack of water or nutrients.

Table 1: Example of an 8th grade science question with a justification for the correct answer. Note the lack of direct lexical overlap present between the justification and the correct answer, demonstrating the difficulty of the task of finding justifications using traditional distant supervision methods.

One approach to interpreting complex models is to make use of human-interpretable information generated by the model to gain insight into what the model is learning. This can be an intermediate representation used by the model, as with the model-generated text spans of Lei et al. (2016), that serve as input to another classification network. By learning these intermediate representations end-to-end with a downstream task, they are optimized to correlate with what the model learns is discriminatory for the task, and they can be evaluated against what a human would consider to be important. Here we apply this general framework for model interpretability to QA.

In this work, we focus on answering multiple-choice science exam questions (Clark (2015); see example in Table 1). This domain is challenging as: (a) approximately 70% of science exam question shave been shown to require complex forms of inference to solve (Clark et al., 2013; Jansen et al., 2016), and (b) there are few structured knowledge bases to support this inference. Within this domain, we propose an approach that learns to both select and explain answers, when the only supervision available is for which answer is correct (but not how to explain it). Intuitively, our approach chooses the justifications that provide the most help towards ranking the correct answers higher than incorrect ones. More formally, our neural network approach alternates between using the current model with max-pooling to choose the highest scoring justifications for correct answers, and optimizing the answer ranking model given these justifications. Crucially, these reranked texts serve as our human-readable answer justifications, and by examining them, we gain insight into what the model learned was useful for the QA task.

The specific contributions of this work are:

1. We propose an end-to-end neural method for learning to answer questions and select a high-quality justification for those answers. Our approach re-ranks free-text answer justifications without the need for structured knowledge bases. With supervision only for the correct answers, we learn this re-ranking through a form of distant supervision – i.e., the answer ranking supervises the justification re-ranking.

2. We investigate two distinct categories of features in this "little data" domain: explicit features, and learned representations. We show that, with limited training, explicit features perform far better despite their simplicity.

3. We demonstrate a large (+9%) improvement in generating high-quality justifications over a strong information retrieval (IR) baseline, while maintaining near state-of-the-art performance on the multiple-choice science-exam QA task, demonstrating the success of the end-to-end strategy.

2 Related work

In many ways, deep learning has become the canonical example of the "black box" of machine learning and many of the approaches to explaining it can be loosely categorized into two types: approaches that try to interpret the parameters themselves (e.g., with visualizations and heat maps (Zeiler and Fergus, 2014; Hermann et al., 2015; Li et al., 2016), and approaches that generate a human-interpretable metric that is ideally correlated with what is being learned inside the model (e.g., Lei et al. (2016)). Our approach falls into the latter type – we use our model's reranking of human-readable justifications to give us insight into what the model considers informative for answering questions. This allows us to see where we do well (Section 6.2), and where we can improve (Section 6.3).

Deep learning has been successfully applied to many recent QA approaches and related tasks (Bordes et al., 2015; Hermann et al., 2015; He and Golub, 2016; Dong et al., 2015; Tan et al., 2016, inter alia). However, large quantities of data are needed to train the millions of parameters often contained in these models. Recently, simpler model architectures have been proposed that greatly reduce the number of parameters while maintaining high performance (e.g., Iyyer et al., 2015; Chen et al., 2016; Parikh et al., 2016). We take inspiration from this trend and propose a simple neural architecture for our task to offset the limited available training data.

Another way to mitigate sparse training data is to include higher-level explicit features. Like Sachan et al. (2016), we make use of explicit features alongside features from distributed representations to capture connections between questions, answers, and supporting text. However, we use a simpler set of features and while they use structured and semi-structured knowledge bases, we use only free-text.

Our approach to learning justification reranking end-to-end with answer selection is similar to the Jansen et al. (2017) latent reranking perceptron, which also operates over free text. However, our approach does not require decomposing the text into an intermediate representation, allowing our technique to more easily extend to larger textual knowledge bases.

The way we have formulated our justification selection (as a re-ranking of knowledge base sentences) is related to, but distinct from the task of answer sentence selection (Wang and Manning, 2010; Severyn and Moschitti, 2012, 2013; Severyn et al., 2013; Severyn and Moschitti, 2015; Wang and Nyberg, 2015, inter alia). Answer sentence selection is typically framed as a fully or semi-supervised task for factoid questions, where a correctly selected sentence fully contains the answer text. Here, we have a variety of questions, many of which are non-factoid. Additionally, we have no direct supervision for our justification selec-

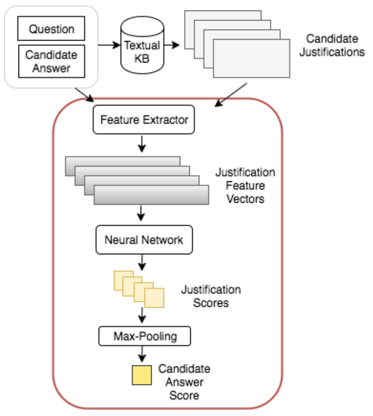

Figure 1: Architecture of our question answering approach. Given a question, candidate answer, and a free-text knowledge base as inputs, we generate a pool of candidate justifications, from which we extract feature vectors. We use a neural network to score each and then use max-pooling to select the current best justification. This serves as the score for the candidate answer itself. The red border indicates the components that are trained online.

tion (i.e., no labels as to which sentences are good justifications for our answers), motivating our distant supervision approach where the performance on our QA task serves as supervision for selecting good justifications. Further, we are not actually looking for sentences that *contain* the answer choice, as with answer sentence selection, but rather sentences which close the ”lexical chasm” (Berger et al., 2000) between question and answer (demonstrated in the example in Table 1).

3 Approach

One of the primary difficulties with the explainable QA task addressed here is that, while we have supervision for the correct answer, we do not have annotated answer justifications. Here we tackle this challenge by using the QA task performance as supervision for the justification reranking, allowing us to learn to choose both the correct answer and a compelling, human-readable justification for that answer.

Additionally, similar to the strategy Chen and Manning (2014) applied to parsing, we combine representation-based features with explicit features that capture additional information that is difficult to model through embeddings, especially with limited training data.

The architecture of our approach is summarized in Figure 1. Given a question and a candidate answer, we first query an textual knowledge base (KB) to retrieve a pool of potential justifications for that answer candidate. For each justification, we extract a set of features designed to model the relations between questions, answers, and answer justifications based on word embeddings, lexical overlap with the question and answer candidate, discourse, and information retrieval (IR) (Section 4.2). These features are passed into a simple neural network to generate a score for each justification, given the current state of the model. A final max-pooling layer selects the top-scoring justification for the candidate answer and this max score is used also as the score for the answer candidate. The system is trained using correct-incorrect answer pairs with a pairwise margin ranking loss objective function to enforce that the correct answer be ranked higher than any of the incorrect answers.

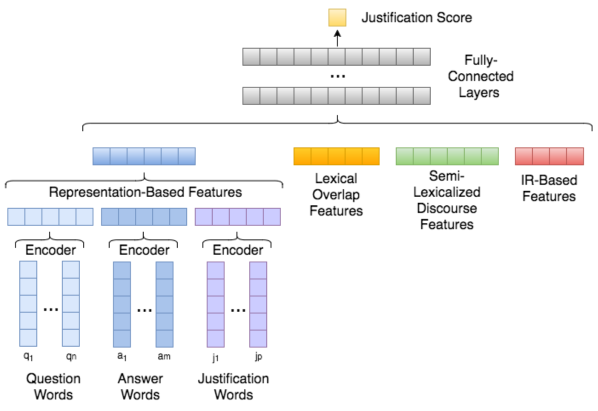

Figure 2: Detailed architecture of the model’s scoring component. The question, candidate answer, and justification are encoded (by summing their word embeddings) to create vector representations of each. These representations are combined in several ways to create a set of representation-based similarity features that are concatenated to additional explicit features capturing lexical overlap, discourse and IR information and fed into a feed-forward neural network. The output layer of the network is a single node that represents the score of the justification candidate.

With this end-to-end approach, the model learns to select justifications that allow it to correctly answer questions. We hypothesize that this approach enables the model to indirectly learn to choose justifications that provide good explanations as to why the answer is correct. We empirically test this hypothesis in Section 6, where we show that indeed the model learns to correctly answer questions, as well as to select high-quality justifications for those answers.

4 Model and Features

Our approach consists of three main components: (a) the retrieval of a pool of candidate answer justifications (Section 4.1); (b) the extraction of features for each (Section 4.2); and (c) the scoring of the answer candidate itself based on this pool of justifications (Section 4.3). The architecture of this latter scoring component is shown in Figure 2.

4.1 Candidate Justification Retrieval

The first step in our process is to use standard information retrieval (IR) methods to retrieve a set of candidate justifications for each candidate answer to a given question. To do this, we build a bag-of-words (BOW) query using the content lemmas for the question and answer candidate, boosting the answer lemmas to have four times more weight[1]. We used Lucene[2] with a *tf-idf* based scoring function to return the top-scoring documents from the KB. Each of these indexed documents consists of a single sentence from our corpora, and serves as one potential justification.

4.2 Feature Extraction

For each retrieved candidate justification, we extract a set of features based on (a) distributed representations of the question, candidate answer, and justification terms; (b) strict lexical overlap; (c) discourse relations present in the justification; and (d) the IR scores for the justification.

Representation-based features (Emb): To model the similarity between the text of each question (Q), candidate answer (A), and candidate justification (J), we include a set of features that utilize distributed representations of the words found in each. First we encode each by summing the vectors for each of their words.[3]. We then compute $sim(Q, A)$, $sim(Q, J)$, and $sim(A, J)$ using cosine similarity. Using another vector representation of only the *unique* words in the justification, i.e., the words that do not occur in either the question or the candidate answer, we also compute $sim(Q, uniqueJ)$ and $sim(A, uniqueJ)$.

To create a feature which captures the relationship between the question, answer, *and* justification, we take inspiration from TransE, a popular relation extraction framework (Bordes et al., 2013). TransE is based on the premise that if two entities, e_1 and e_2 are related by a relation r, then a mapping into k dimensions, $m(x) \in \mathbb{R}^k$ can be learned such that $m(e_1) + m(r) \approx m(e_2)$. Here, we modify this intuition for QA by suggesting that given the vectorized representations of the question, answer candidate, and justification above, $Q + J \approx A$, i.e., a question combined with a strong justification will point towards an answer. Here we model this as an explicit feature, the euclidean distance between $Q + J$ and A, and hypothesize that as a consequence the model will learn to select passages that maximize the quality of the justifications. This makes a total of six features based on distributed representations.

Lexical overlap features (LO): We additionally characterize each justification in terms of a simple set of explicit features designed to capture the size of the justification, as well as the lexical overlap (and difference) between the justification and the question and answer candidate. We include these five features: the proportion of question words, of answer words, and of the combined set of question and answer words that also appear in the justification; the proportion of justification words that do not appear in either the question or the answer; and the length of the justification in words.[4]

Semi-Lexicalized Discourse features (lexDisc): These features use the discourse structure of the justification text, which has been shown to be useful for QA (Jansen et al., 2014; Sharp et al., 2015; Sachan et al., 2016).

We use the discourse parser of Surdeanu et al. (2015) to fragment the text into elementary discourse units (EDUs) and then recursively connect neighboring EDUs with binary discourse relations. For each of the 18 possible relation labels, we create a set of semi-lexicalized discourse features that indicate the presence of a given discourse relation as well as whether or not the head and modifier texts contain words from the question and/or the answer.

For example, for the question *Q: What makes water a good solvent...? A: strong polarity*, with a discourse-parsed justification [*Water is an efficient solvent*]$_{e1}$ [*because of this polarity.*]$_{e2}$, we create the semi-lexicalized feature *Q_cause_A*, because there is a *Cause* relation between EDUs $e1$ and $e2$, $e1$ overlaps with the question, and $e2$ overlaps with the answer. Since there are 18 possible discourse relation labels, and the prefix and suffix can be any of *Q, A, QA* or *None*, this creates a set of 288 indicator features.

[1]We empirically found this answer term boosting to ensure retrieval of documents which were relevant to the particular answer candidate.

[2]`https://lucene.apache.org`

[3]While this BOW approach is not ideal in many ways, it performed equivalently to far more complicated approaches such as LSTMs and GRUs, also noted by (Iyyer et al., 2015), likely due to the limited training data in this domain.

[4]We normalized this value by the maximum justification length.

IR-based features (IR^{++}): Finally, we also use a set of four IR-based features which are assigned at the level of the answer candidate (i.e., these features are identical for each of the candidate justifications for that answer choice). Using the same query method as described in Section 4.1, for each question and answer candidate we retrieve a set of indexed documents. Using the *tf-idf* based retrieval scores of these returned documents, $s(d_i)$ for $d_i \in D$, we rank the answer candidates using two methods:

- by the maximum retrieved document score for each candidate, and
- by the weighted sum of all retrieved document scores[5]:
$$\sum_{d_i \in D} \frac{1}{i} s(d_i) \tag{1}$$

We repeat this process using an unboosted query as well, for a total of four rankings of the answer candidates. We then use these rankings to make a set of four reciprocal rank features, IR$_0^{++}$, ..., IR$_3^{++}$, for each answer candidate (i.e., IR$_0^{++} = 1.0$ for the top-ranked candidate in the first ranking, IR$_0^{++} = 0.5$ for the next candidate, etc.)

4.3 Neural Network

As shown in Figure 2, the extracted features for each candidate justification are concatenated and passed into a fully-connected feed-forward neural network (NN). The output layer is a single node representing the justification score. We then use max-pooling over these scores to select the current best justification for the answer candidate, and use its score as the score for the answer candidate itself. For training, the correct answer for a given question is paired with each of the incorrect answers, and each are scored as above. We compute the pair-wise margin ranking loss for each training pair:

$$L = \max(0, m - F(a^+) + F(a^-)) \tag{2}$$

where $F(a^+)$ and $F(a^-)$ are the model scores for a correct and incorrect answer candidate and m is the margin, and backpropagate the gradients. At testing time, we use the trained model to score each answer choice (again using the maximum justification score) and select the highest-scoring.

As we are interested in not only correctly answering questions, but also selecting valid justification for those answers, we keep track of the scores of *all* justifications and use this information to return the top k justifications for each answer choice. These are evaluated along with the answer selection performance in Section 6.

5 Experiments

5.1 Data and Setup

We evaluated our model on the set of 8th grade science questions that was provided by the Allen Institute for Artificial Intelligence (AI2) for a recent Kaggle challenge. The training set contained 2,500 question, each with 4 answer candidates. For our test set, we used the 800 publicly-released questions that were used as the validation set in the actual evaluation.[6] We tuned our model architectures and hyper-parameters on the training data using five-fold cross-validation (training on 4 folds, validating on 1). During testing, we froze the model architecture and all hyperparameters and re-trained on all the training data, setting aside a random 15% of training questions to facilitate early stopping.

5.2 Baselines

In addition to previous work, we compare our model against two strong IR baselines:

- **IR Baseline:** For this baseline, we rank answer candidates by the maximum *tf.idf* document retrieval score using an unboosted query of question and answer terms (see Section 4.1 for retrieval details).
- **IR^{++}:** This baseline uses the same architecture as the full model, as described in Section 4.3, but with only the IR^{++} feature group.

5.3 Corpora

For our pool of candidate justifications (as well as the scores for our IR baselines) we used the corpora that were cited as being most helpful to the top-performing systems of the Kaggle challenge. These consisted of short, flash-card style texts gathered from two online resources: about 700K sentences from StudyStack[7] and 25K sentences from Quizlet[8]. From these corpora, we use the top 50 sentences retrieved by the IR model as our set of candidate justifications. All of our corpora were annotated using using the Stanford

[5]Weighted sum was based on the IR scores used in the winning Kaggle system from user Cardal (`https://github.com/Cardal/Kaggle_AllenAIscience`)

[6]The official testing dataset is not publicly available.

[7]`https://www.studystack.com/`

[8]`https://quizlet.com/`

CoreNLP toolkit (Manning et al., 2014), the dependency parser of Chen and Manning (2014), and the discourse parser of Surdeanu et al. (2015).

While our model is able to learn a set of embeddings, we found performance was improved when using pre-trained embeddings, and in this low-data domain, fixing these embeddings to not update during training substantially reduced the amount of model over-fitting. In order to pretrain domain-relevant embeddings for our vocabulary, we used the documents from the StudyStack and Quizlet corpora, supplemented by the newly released Aristo MINI corpus (December 2016 release)[9], which contains 1.2M science-related sentences from various web sources. The training was done using the `word2vec` algorithm (Mikolov et al., 2010, 2013) as implemented by Levy and Goldberg (2014), such that the context for each word in a sentence is composed of all the other words in the same sentence. We used embeddings of size 50 as we did not see a performance improvement with higher dimensionality.

5.4 Model Tuning

The neural model was implemented in Keras (Chollet, 2015) using the Theano (Theano Development Team, 2016) backend. For our feed-forward component, we use a shallow neural network that we lightly tuned to have a single fully-connected layer containing 10 nodes, glorot uniform initialization, a $tanh$ activation, and an L2-regularization of 0.1. We trained with the RMSProp optimizer (Tieleman and Hinton, 2012), a learning rate of 0.001, 100 epochs, a batch size of 32, and early stopping with a patience of 5 epochs. Our loss function used a margin of 1.0.

We experimented with burn-in, i.e., using the best justification chosen by the IR model for the first mini-batches, but found that models without burn-in performed better, indicating that the model benefited from being able to select its own justification.

6 Results

Rather than seeking to outperform all other systems at selecting the correct answer to a question, here we aimed to construct a system system that can produce substantially better justifications for why the answer choice is correct to a human user, without unduly sacrificing accuracy on the answer selection task. Accordingly, we evaluate our system both in terms of it's ability to correctly answer questions (Section 6.1), as well as provide high-quality justifications for those answers (6.2). Additionally, we perform an error analysis (Section 6.3), taking advantage of the insight the reranked justifications provide into what the model is learning.

[9] `http://allenai.org/`

#	Model	P@1 Val	P@1 Test
1	Random	25	25
2	IR Baseline	47.2	47
3	IR^{++}	50.7**	36.35
4	Iyyer et al. (2015)	–	32.52
5	Khot et al. (2017)	–	46.17
6	Our approach w/o IR	50.54*	48.66
7	Our approach	**54.0****††	**53.3****†

Table 2: Performance on the AI2 Kaggle questions, measured by precision-at-one (P@1). *s indicate that the difference between the corresponding model and the IR baseline is statistically significant (* indicates $p < 0.05$ and ** indicates $p < 0.001$) and †s indicate significance compared to IR^{++}, All significance values were determined through a one-tailed bootstrap resampling test with 100,000 iterations.

Ablated Model	P@1 Val
IR^{++} + LO	53.4**††
IR^{++} + LO + lexDisc	53.6**††
Full Model (IR^{++} + LO + lexDisc + Emb)	**54.0****††

Table 3: Ablation of feature groups results, measured by precision-at-one (P@1) on validation data. Significance is indicated as in Table 2.

6.1 QA Performance

We evaluated the accuracy of our system as well as the baselines on the held-out 800 set of test questions. Performance, measured in precision at 1 (P@1)(Manning et al., 2008), is shown in Table 2 for both the validation (i.e., cross validation on training) and test partitions. Because NNs are sensitive to initialization, each experimental result shown is the average performance across five runs, each using different random seeds.

The best performing baseline on the validation data was a model using only IR^{++} features (line 3), but its performance dropped substantially when evaluated on test due to the failure of several random seed initializations to learn. For this reason, we assessed significance of our model combinations with respect to both the IR baseline as well as the IR^{++} (indicated by * and †s, respectively).

Our full model that combines IR^{++}, lexical overlap, discourse, and embeddings-based features, has a P@1 of 53.3% (line 7), an absolute gain of 6.3% over the strong IR baseline despite using the same background knowledge.

Comparison to Previous Work: We compared our performance against another model that achieves state of the art performance on a different set of 8th grade science questions, TUPLEINF(T+T') (Khot et al., 2017). TUPLEINF(T+T') uses Integer Linear Programming to find support for questions via tuple representations of KB sentences[10]. On our test data, TUPLEINF(T+T') achieves 46.17% P@1 (line 5). As this model is independent of an IR component, we compare its performance against our full system without the IR-based features (line 6), whose performance is 48.66% P@1, an absolute improvement of 2.49% P@1 (5.4% relative) despite our unstructured text inputs and the far smaller size of our knowledge base (three orders of magnitude).

Sachan et al. (2016) also tackle the AI2 Kaggle question set with an approach that learns alignments between questions and structured and semi-structured KB data. They use only the training questions (splitting them into training, validation, and testing partitions), supplemented by questions found in online study guides, and report an accuracy of 47.84%. By way of a loose comparison (since we are evaluating on different data partitions), our model has approximately 5% higher performance despite our simpler set of features and unstructured KB.

We also compare our model to our implementation of the basic Deep-Averaged Network (DAN) Architecture of Iyyer et al. (2015). We used the same 50-dimensional embeddings in both models, so with the reduced embedding dimension, we reduced the size of each of the DAN dense layer to 50 as well. For simplicity, we also did not implement their word-dropout, a feature that they reported as providing a performance boost. Using this implementation, the performance on the test set was 31.50% P@1. To help with observed overfitting, we tried removing the dense layers and received a small boost to 32.52% P@1 (line 4). The lower performance of their model, which relies exclusively on latent representations of the data, underscores the benefit of including explicit features alongside latent features in a deep-learning approach for this domain.

In comparison to other systems that competed in the Kaggle challenge, our system comes in in 7th place out of 170 competitors (top 4%).[11] Compared with the systems which disclosed their methods, we use a subset of their corpora and substantially less hyperparameter tuning, and yet we achieve competitive results.

[10]Notably, one portion of the tuple KB used was constructed based on a different 8th grade question set than the one we use here.

Question
Q: Scientists use ice cores to help predict the impact of future atmospheric changes on climate. Which property of ice cores do these scientists use? **A:** The composition of ancient materials trapped in air bubbles

Rating	Example Justification
Good	Ice cores: cylinders of ice that scientist use to study trapped atmospheric gases and particles frozen with in the ice in air bubbles
Half	Ice core: sample from the accumulation of snow and ice over many years that have recrystallized and have trapped air bubbles from previous time periods
Topical	Vesicular texture formation [has] trapped air bubbles.
Off-topic	Physical change: change during which some properties of material change but ...

Table 4: Example justifications from the our model and their associated ratings.

Feature Ablation: To evaluate the contribution of the individual feature groups, we additionally performed an ablation experiment (see Table 3). Each of our ablated models performed significantly better than the IR baseline on the validation set, including our simplest model, IR^{++}+LO.

6.2 Justification Performance

One of our key claims is that our approach addresses the related, but more challenging problem of performing *explainable* question answering, i.e., providing a high-quality, compelling justification for the chosen answer. To evaluate this claim, we evaluated a random set of 100 test questions that both the IR baseline and our full system answered correctly. For each question, we assessed the quality of each of the top five justifications. For IR, these were the highest-scoring retrieved documents, and for our system, these were the top-scoring justifications as re-ranked by our model. Following the methodology of Jansen et al. (2017), each justification received a rating of either *Good* (if the connection between the ques-

[11]Based on the public leaderboard (`https://www.kaggle.com/c/the-allen-ai-science-challenge/leaderboard`). The best scoring submission had an accuracy of 59.38%. Note that for the systems that participated, this set served as *validation* while for us it was test, and thus it is likely that these scores are slightly overfitted to this dataset, but for us it was blind. As such this is a conservative comparison, and in reality the difference is likely to be smaller.

Model	Good@1	Good@5	NDCG@5
IR Baseline	0.52	0.64	0.55
Our Approach	**0.61**	**0.74**	**0.62****

Table 5: Percentage of questions that have at least one *good* justification within the top 1 (Good@1) and the top 5 (Good@5) justifications, as well as the normalized discounted cumulative gain at 5 (NDCG@5) of the ranked justifications. Significance indicated as in Table 2.

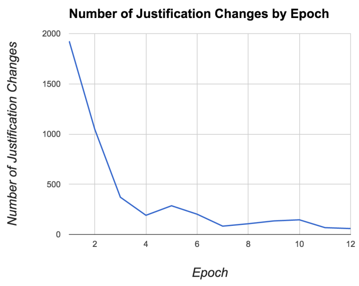

Figure 3: Number of questions for which our complete model chooses a new justification at each epoch during training. While this is for a single random seed, we see essentially identical graphs for each random initialization.

tion and correct answer was fully covered), *Half* (if there was a missing link), *Topical* (if the justification was simply of the right topic), or *Off-Topic* (if the justification was completely unrelated to the question). Examples of each rating are provided in Table 4.

Results of this analysis are shown using three evaluation metrics in Table 5. The first two columns show the percentage of questions which had a *Good* justification at position 1 (Good@1), and within the top 5 (Good@5). Note that 61% of the top-ranked justifications from our system were rated as *Good* as compared to 52% from the IR baseline (a gain of 9%), despite the systems using identical corpora.

We also evaluated the justification ratings using normalized discounted cumulative gain at 5 (NDCG@5) (as formulated in Manning et al. (2008), p.163), where we assigned *Good* justifications a gain of 3.0, *Half* a gain of 2.0, *Topical* a gain of 1.0, and *Off-Topic* a gain of 0.0. With this formulation, our system had a NDCG@5 of 0.62 while the IR baseline had a significantly lower NDCG@5 of 0.55 ($p < 0.001$), shown in the third column of Table 5.

Contribution of Learning to Rerank Justifications: The main assertion of this work is that through learning to rank answers and justifications for those answer candidates in an end-to-end manner, we both answer questions correctly and provide compelling justifications as to why the answer is correct. To confirm that this is the case, we also ran a version of our system that does not rerank justifications, but uses the top-ranked justification retrieved by IR. This configuration dropped our performance on test to 48.7% P@1, a decrease of 4.6%, and we additionally lose all justification improvements from our system (see Section 6.2), demonstrating that learning this reranking is key to our approach.

Additionally, we tracked the number of times a new justification was chosen by the model as it trained. We found that our system converges to a stable set of justifications during training, shown in Figure 3.

6.3 Error Analysis

We performed an error analysis of 30 incorrectly answered questions, examining the top 5 justifications returned for both the correct and chosen answers of each. Notably, 50% of the questions had one or more *Good* justifications. The most common form of error (53.3%) was due to the system's preference for short justifications with a large degree of lexical overlap with the question and answer choice, suggesting the system has learned that generally many unmatched words are indicative of an incorrect answer. The second largest source of errors (43.3%) came from questions requiring complex inference (causal, process, quantitative, or model-based reasoning), demonstrating the difficulty of the question set and the need for systems that can robustly handle a variety of question types. Aside from these main groups, there were some smaller trends including KB noise, and our system's lack of using word order and recognizing negation.[12]

7 Conclusion

Here we propose an end-to-end question answering (QA) model that learns to correctly answer questions as well as provide compelling, human-readable justifications for its answers, despite not having access to labels for justification quality. We do this by using the question answering task as a form of distant supervision for learning justification re-ranking. We show that our accuracy and justification quality are significantly better than a strong IR baseline, while maintaining near state-of-the-art performance for the answer selection task as well.

[12]A much more detailed error analysis is available, and if this paper is accepted will be included.

References

Adam Berger, Rich Caruana, David Cohn, Dayne Freytag, and Vibhu Mittal. 2000. Bridging the lexical chasm: Statistical approaches to answer finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research & Development on Information Retrieval*. Athens, Greece.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR* abs/1506.02075.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

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)*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Empirical MNLP*. pages 740–750.

F. Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Peter Clark. 2015. Elementary school science and math tests as a driver for AI: take the Aristo challenge! In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*. AAAI Press, pages 4019–4021.

Peter Clark, Philip Harrison, and Niranjan Balasubramanian. 2013. A study of the knowledge base requirements for passing an elementary science test. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*. AKBC'13, pages 37–42.

Mark W Craven and Jude W Shavlik. 1996. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems* pages 24–30.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of Association for Computational Linguistics*. pages 260–269.

Xiaodong He and David Golub. 2016. Character-level question answering with attention. In *EMNLP*.

Karl Moritz Hermann, Tomáš Kočiský, 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 (NIPS)*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics*.

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 *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 2956–2965.

Peter Jansen, Rebecca Sharp, Mihai Surdeanu, and Peter Clark. 2017. Framing qa as building and ranking intersentence answer justifications. *Computational Linguistics* .

Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. In *Proceedings of Association for Computational Linguistics (ACL)*.

Been Kim, Julie A. Shah, and Finale Doshi-Velez. 2015. Mind the gap: A generative approach to interpretable feature selection and extraction. In *NIPS*.

Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2016. Rationalizing neural predictions. In *EMNLP*.

Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics* 9(3):1350–1371.

O. Levy and Y. Goldberg. 2014. Dependency-based word embeddings. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 302–308.

Jiwei Li, Xinlei Chen, Eduard H. Hovy, and Daniel Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *HLT-NAACL*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *HLT-NAACL Demos*.

Mrinmaya Sachan, Avinava Dubey, and Eric P Xing. 2016. Science question answering using instructional materials. In *The 54th Annual Meeting of the Association for Computational Linguistics*. page 467.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. In *SIGIR*.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning adaptable patterns for passage reranking. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CoNLL)*.

Rebecca Sharp, Peter Jansen, Mihai Surdeanu, and Peter Clark. 2015. Spinning straw into gold: Using free text to train monolingual alignment models for non-factoid question answering. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 231–237.

Mihai Surdeanu, Thomas Hicks, and Marco A. Valenzuela-Escárcega. 2015. Two practical rhetorical structure theory parsers. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL): Software Demonstrations*.

Ming Tan, Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *ACL*.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. *ACL, July* .

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *COLING*.

Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *ECCV*.