Simple and Effective Question Answering with Recurrent Neural Networks

Ferhan Ture and Oliver Jojic
Comcast Labs, Washington, DC 20005
{ferhan.ture, oliver.jojic}@cable.comcast.com

Abstract

First-order factoid question answering assumes that the question can be answered by a single fact in a knowledge base (KB). While this does not seem like a challenging task, many recent attempts that apply either complex linguistic reasoning or deep neural networks achieve 35%–65% accuracy on benchmark sets. Our approach formulates the task as two machine learning problems: detecting the entities in the question, and classifying the question as one of the relation types in the KB. Based on this assumption of the structure, our simple yet effective approach trains two recurrent neural networks to outperform state of the art by significant margins — relative improvement reaches 16% for WebQuestions, and surpasses 38% for SimpleQuestions.

1 Introduction

Two major approaches have been dominating the Question Answering (QA) literature in the last decade. Some work have heavily invested on converting the question into a linguistically motivated representation (e.g., using syntactic and/or semantic parsing), which simplifies the problem into finding the answer that best fits this representation (Buscaldi et al., 2010; Bilotti et al., 2007; Bilotti et al., 2010; Berant et al., 2013; Reddy et al., 2014). For factoid QA, where the answer to the question is an indisputable fact, and all known facts are stored in a knowledge base (KB) in some structured form — in this case, the question can be represented in a form that is consistent with the KB.

The recent success of deep learning has tempted many researchers to explore more language-agnostic approaches, relying on automatically learning representations in a data-driven manner. With an increasing amount of access to computational resources and data, neural networks have outperformed traditional approaches on a variety of natural language processing (NLP) tasks (Irsoy and Cardie, 2014; Kim, 2014; Chen and Manning, 2014), including QA (Hsu et al., 2016; Yang et al., 2014; Dong et al., 2015; Bordes et al., 2015).

We argue that both linguistically-motivated and learning-based approaches have complementary benefits, and therefore propose a novel method for factoid QA that can be considered as a hybrid: given a knowledge base (KB), a recurrent neural network (RNN) is trained to automatically convert a given question into a structured query (Section 3a), which is then queried through the KB in order to retrieve the most relevant answer(s) (Section 3b).

Our assumption is that a question can be represented as a pair: (a) the entity the question is about, and (b) the aspect of the entity we are interested in. If a question fits into this assumption, we refer to it as a first-order question. While this sounds similar to the “Single Supporting Fact” task in (Weston et al., 2015), for which they report 100% accuracy, the problem becomes much more complex as the knowledge base grows to a realistic size, since the level of ambiguity increases drastically with a larger vocabulary and list of entities. This is the reason why we focus on well-studied, large-scale QA tasks like Free917 (Cai and Yates, 2013), WebQuestions (Be-
rant et al., 2013) (WQ), and SimpleQuestions (Bordes et al., 2015) (SQ), that rely on Freebase as a realistic knowledge base. Given that the state-of-the-art accuracy is around 45% for WQ and 64% for SQ, this is far from a solved problem, and there is still plenty of room for improvement.

We build our novel approach based on the hypothesis that majority of questions in these tasks are first-order factoid questions. Results on two most recent benchmark tasks, SimpleQuestions and WebQuestions, strongly confirm our hypothesis. The accuracy of our approach indicates major improvements over the state of the art — relative improvement reaches 16% for WQ, and surpasses 38% for SQ.

2 Related work

We focus on QA over structured knowledge bases. In this scenario, if the question and knowledge can be formulated in a compatible way, reasoning can be performed to determine which fact in the knowledge base answers a given question. Many methods have been proposed for parsing questions, such as a pattern-based question analyzer (Buscaldi et al., 2010), combination of syntactic parsing and semantic role labeling (Bilotti et al., 2007; Bilotti et al., 2010), as well as lambda calculus (Berant et al., 2013) and combinatory categorial grammars (CCG) (Reddy et al., 2014). The CCG-based semantic parser, trained on ClueWeb sentences annotated with corresponding Freebase facts, achieves 41% accuracy on the WebQuestions dataset.

The major downside of these approaches is the reliance on linguistic resources and/or heuristics, which also means that the methods are language- and/or domain-specific, and therefore do not generalize well. Even though Reddy et al. (2014) claim that their approach does not require as much supervision as prior work, it still relies on many English-specific heuristics and hand-crafted features. Also, their most accurate model uses a corpus of paraphrases to generalize to linguistic diversity. Also, a practical weakness of linguistic parsers is that they are usually too slow to provide sufficiently low latency in an online QA setting.

In contrast, our RNN-based parser detects entities in the question with very high accuracy (≈90%) and low latency (< 50ms). The only required resources are a set of questions with entities tagged and a word embedding. The latter can be easily trained for any language or domain in an unsupervised fashion, given a large text corpus without annotations (Mikolov et al., 2013; Pennington et al., 2014). The former is a relatively simpler annotation task that exists for many languages and domains, and it can also be synthetically generated. One of the strengths of our approach is to combine word embeddings with deep learning to learn robust semantic representations. While many researchers have explored this space for general NLP tasks (Collobert et al., 2011), such as named entity recognition (Lu et al., 2015; Hammerton, 2003), sequence labeling (Graves, 2008; Chung et al., 2014), part-of-speech tagging (Huang et al., 2015; Wang et al., 2015), chunking (Huang et al., 2015), we are not aware of the use of RNNs to parse questions for QA. Among the many recent advancements in applying deep learning techniques to QA, a general trend is to construct parallel neural networks for modeling a joint probability distribution for question-answer pairs (Hsu et al., 2016; Yang et al., 2014; He et al., 2015; Mueller and Thyagarajan, 2016). Memory Networks (Bordes et al., 2015) are also based on a similar principle, trying to discover supporting facts by matching the question to the compiled memory. This is probably needed for general-purpose question answering and sentence similarity, where one cannot make assumptions about the structure of the input or knowledge. However, as noted in Section 1, first-order factoid questions can be represented by an entity and a relation type, and the answer is usually stored in a structured knowledge base. Dong et al. (2015) similarly make the assumption that each question contains a target entity, and search for the best answer at most two hops away in the KB. However, they do not propose how to obtain the target entity, as they assume it is provided as part of the task. Bordes et al. (2014) take advantage of the KB structure by projecting entities, relations, and subgraphs into the same latent space.

In addition to parsing the question to find the target entity, we mentioned that the other key information is the relation type corresponding to the question. Many researchers have shown that classifying the question into one of the pre-defined types, either based on patterns (Zhang and Lee, 2003) or a trained
support vector machine (Buscaldi et al., 2010), improves QA accuracy. More recently, Iyyer et al. demonstrated the power of RNNs by treating the QuizBowl task as a classification problem (2014). However, the Jeopardy-like task is substantially different than the factoid QA we intend to tackle: every answer is assumed to be one of the thousand entities, which does not hold for many questions (e.g., “How old is Tom Hanks”), and the number of entities in a realistic knowledge base is orders of magnitude higher (~1000 vs. millions). Unlike the paragraph-long questions in the QuizBowl dataset used in (Iyyer et al., 2014), the typical factoid question contains a handful of words. Nonetheless, our RNN-based classifier is impressively accurate at determining which relation type in Freebase (amongst more than 2000 choices) a question corresponds to.

3 Approach
We present a novel approach to answer first-order factoid questions. In factoid question answering, we assume that the answer to the question can be determined objectively. More specifically, we assume that the answer to the question exists in the provided knowledge base. First-order questions further require that the answer can be found in a single fact in the knowledge base.

Even though (Bordes et al., 2015) refer to first-order factoid questions as simple questions, current approaches still struggle in accurately answering them (e.g., state-of-the-art accuracy is around 64% and 45% on the SimpleQuestions and WebQuestions datasets). Two main challenges contribute to this: (a) the same question can be asked in different syntactic and lexical varieties, and (b) there are many entities in a typical knowledge base, making resolution a hard task. Ambiguity might be inherent in the question — e.g., in “When did Kennedy die”, it is unclear which person is being referred to as “Kennedy”.

(a) From Question to Structured Query. Our approach is based on a knowledge base, containing a large set of facts, each one representing a binary [subject, relation, object] relationship. Since we assume first-order questions, the answer can be retrieved from a single fact. For instance, “How old is Tom Hanks?” can be answered by the fact:

[Tom Hanks, born_on, 7/9/1956]

The main idea is to represent the semantics of a first-order factoid question by converting the input question into a structured query with two slots. For example, the question above would be converted to:

{entity: Tom Hanks, relation: born_on}

Once a query is formulated, QA is reduced to a search problem (i.e., retrieving the fact in the knowledge base that best matches the query). Section 3b describes that part of our approach. The rest of this section describes how we construct the query.

The process of converting a natural-language question to a structured query can be modularized into two machine learning problems, namely entity detection and relation prediction. In the former, the objective is to tag whether each question word is part of an entity. In the latter, the objective is to classify the question into one of the K relation types.

We modeled both problems using recurrent neural networks (RNN) (Elman, 1990). Providing a detailed description of RNNs is out of the scope of this paper, but we will give a brief overview in the context of question representation. One of the most important aspects of NLP over sentence-level text (e.g., QA) is to model dependencies across words in the question. This is usually satisfied by engineering features that describe the local context (e.g., whether the word to the left is a verb, the first two letters of the first word in the sentence, etc.). While these approaches perform well in certain cases, they depend on hand-crafting the right set of features, requiring substantial effort and varying based on domain, task, and even dataset.

An RNN naturally models dependencies among all words in the input sequence (i.e., question) without any feature engineering. Certain variations of RNNs, such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRU) (Cho et al., 2014a), also learn contextual features that are useful for the task, while learning to ignore irrelevant parts. For example, the model might learn that the first word in a question might be a useful signal for classification, and that stop words are not useful features regardless of where they appear in the question. RNNs’ effectiveness in exploiting wide textual con-
text has been shown in numerous cases (e.g., sentiment analysis (Socher et al., 2013) and machine translation (Cho et al., 2014b)).

Each word in the question passes through an embedding lookup layer $E$, projecting the one-hot vector into a $d$-dimensional vector $x_t$. A recurrent layer combines this representation with the hidden layer from the previous word with the input ($h_{t-1}$; existing memory) and applies a non-linear transform (e.g., ReLu) to compute the hidden layer representation for the current word ($h_t$; updated memory). The hidden representation of the final recurrent layer is projected to the output space of $k$ dimensions ($k$ is the number of classes), and normalized into a probability distribution via soft-max.

$$h_t = \text{ReLU}(W h_{t-1} + U x_t) \quad \text{hidden repr. at } t$$

$$o_t = \text{softmax}(V h_t) \quad \text{output at } t$$

In the entity detection task, each word is classified as either entity or context, so $k$ is 2. In relation prediction, we classify the entire question into one of the many classes ($k$ varies by task, see Section 4), and the hidden layer of the final word is used for this. In other words, given question $q_1 \ldots q_t \ldots q_n$, for entity detection, $o_t$ is a probability distribution that represents the tag prediction for $q_t$; for relation prediction, $o_n$ is a probability distribution represents the class prediction for the entire question. In both cases, parameters are learned via stochastic gradient descent, using categorical cross-entropy as objective. In order to handle variable-length input, we limit the input to $N$ tokens and prepend a special pad word if input has fewer.\(^2\)

An LSTM or GRU is a type of RNN with a slightly more complex transition function. These models contain gates that determine to what degree that the existing memory is forgotten and that the new input is added to the memory. This allows a more powerful model for longer text sequences, while increasing the number of parameters and requiring more training data as a result. Another useful modification is to allow memory to be passed in both directions: left-to-right and right-to-left. These bidirectional RNN units can use information about

(b) Searching the Knowledge Base. In order to make our knowledge base searchable by a given structured query, we built two indexes: An inverted index $I_{\text{entity}}$ maps all $n$-grams of an entity ($n = 1, 2, 3$) to the entity, with an associated BM25 score. We also map the exact text ($n = \infty$) to be able to find exact matches. A graph reachability index $I_{\text{reach}}$ maps each entity node $e$ to all nodes $e'$ that are reachable, with the associated path $p(e, e')$. For the purpose of the current approach, we limit our search to a single hop away, but this index can be easily expanded to support a wider search.

Search for QA works as follows: Given a new question, we run the two RNN models to construct the structured query $q$={entity: $t_e$, relation: $r$}. Next step is to find the intended entity in the KB. Starting with an exact string match (i.e., $t_e$), we query $I_{\text{entity}}$, which returns all entities in the KB that are potential matches. Retrieved entities are added to the candidate set $C$, with associated BM25 relevance scores. Unless there is a reason to terminate earlier, we iterate over values for $n$ in a descending order (i.e., $n \in \{\infty, 3, 2, 1\}$). For each value of $n$, we query $I_{\text{entity}}$ with all $n$-grams in $t_e$, appending the retrieved list to $C$. Early termination happens if (i) $C$ is non-empty, and (ii) $n$ is less than or equal to the number of tokens in $t_e$. The latter criterion is to avoid cases where although we find an exact match, there are also partial matches that might be more relevant (e.g., “jurassic park” has an exact match to the original movie, but we would also like to retrieve “jurassic park II” as a candidate entity).

Once we have a list of candidate entities $C$, we use each candidate node $e_{\text{cand}}$ as a starting point to reach candidate answers. We query $I_{\text{reach}}$ to retrieve all nodes $e_a$ that are reachable from $e_{\text{cand}}$, where the path from is consistent with the predicted relation $r$ (i.e., $r \in p(e_{\text{cand}}, e_a)$). These are added to the candidate answer set $A$. After repeating this process for each entity in $C$, the highest-scored node in $A$ is our best answer to the question.

4 Experimental Setup

Data. Evaluation of our approach was carried out on two QA benchmark datasets: WebQuestions (WQ)
Table 1: Summary of experimental datasets.

|     | Train | Tune | Test (rmvd) | Rel. types | Vocab |
|-----|-------|------|-------------|------------|-------|
| WQ  | 5142  | 1808 | 2032 (200)  | 1181       | 2.9k  |
| SQ  | 75910 | 10845| 21687 (1294)| 1837       | 55k   |

and SimpleQuestions (SQ). Both WQ and SQ are based on the same subset of Freebase, which contains 17.8M million facts, 4M unique entities, and 7523 relation types. Indexes \(I_{\text{entity}}\) and \(I_{\text{reach}}\) are built based on this knowledge base. We first describe the datasets and how they differ in terms of the question collection process and scale.

WebQuestions was constructed by (Berant et al., 2013) as a factoid question answering benchmark set. Question templates were constructed using Google Suggest API, which were then instantiated by asking users on Mechanical Turk to annotate each question with an answer, including the Freebase page from which the answer was found. However, neither entities in the question text nor the relation type are provided. In order to use this dataset in our evaluation, we needed to infer this information from Freebase. This is a non-trivial task that results in noisy labels, so we decided to share the details with the research community.

We started with a post-processed version of WebQuestions, which uses a combination of heuristics and an existing QA system to obtain the following information for each question: all entities mentioned in the question (along with Freebase identifier) and set of relation types that could have resulted in the provided textual answer. Many questions are correctly associated with multiple relation types, each corresponding to a different path to the same answer. We created a separate training instance for each one, and at test time, we accept any of the possible relations as the correct prediction (since all of them result in the same answer).

SimpleQuestions was built by (Bordes et al., 2014) to serve as a larger, more diverse QA dataset. Human annotators were told to create questions from a Freebase fact. Facts were sampled in a way that more frequent relation types had a lower chance of being selected. This ensures a diverse set of questions that are fully compatible with Freebase, and are labeled with corresponding entity and relation type.

Statistics about the datasets are summarized in Table 1. For both datasets, even after the process above, there was a small subset for which we could not identify either an associated relation, or the correct entity. These were removed from the test set, since we cannot evaluate our approach. There is no reason to believe these questions were more difficult than others, so we assume that our accuracy would have been same on the removed portion, and compare against published results accordingly. But even if we assume all of them to be answered incorrectly, our conclusions would not change.

Training. Input length \(N\) was set to the maximum number of tokens across training and validation splits: 36 for SQ and 15 for WQ models. We fixed the embedding layer based on the pre-trained 300-dimensional Google News embedding, since the data size is too small for training embeddings. Out-of-vocabulary words were assigned to a random vector (sampled from uniform distribution). We tried a variety of configurations for the RNN: four choices for the type of RNN layer (GRU or LSTM, bidirectional or not); depth from 1 to 3; and dropout ratio from 0 to 0.5, yielding a total of 48 possible configurations. For each possible setting, we trained the model on the training portion and used the validation portion to avoid over-fitting. Training was terminated if the validation loss did not improve for five epochs. Unsurprisingly, we observed that most learning occurred in the first few epochs, yet the final models usually trained for up to 100 epochs (several hours). After running all 48 experiments, the most optimal setting was selected by micro-average F-score of predicted entities (entity detection) or accuracy (relation prediction) on the validation set. The same process was repeated for both WQ and SQ tasks.

Results

Model selection. The effect of the various parameter settings on model effectiveness can be seen in Figures 1(a) and 1(b). Each figure shows
results on 48 different settings, as evaluated on the validation set of the specified dataset and task. Every line is labeled with naming convention \((Bi)(LSTM|GRU)(1|2|3)\), and each point represents a different drop-out value (i.e., x axis).

LSTM (circle or star) seems to work better for entity detection, and GRU (diamond or square) outperforms on relation prediction. A drop-out beyond 20% became degrading (especially for deeper models), when there was simply too much noise to learn such a complex model. More training examples could have been a remedy for this. Based on these results, we concluded that the optimal model is a 2-layer bidirectional LSTM (BiLSTM2) for entity detection on SQ, and a BiLSTM1 on WQ (10% drop-out in both cases). For relation prediction on both SQ and WQ, the most optimal model is BiGRU2. In the former dataset, a 10% drop-out was optimal, whereas 20% worked better on WQ.

Relation prediction accuracy is artificially low on WQ due to the way we prepared the data: The same input question is labeled as multiple relations, since these had to be inferred from Freebase, and were not originally published as part of the dataset. Even though this results in lower accuracy during training, it does not worsen the learning process, since all of the labels are potentially correct. We will describe how we can take advantage of other information to select the most viable relation at runtime.

|       | WQ     | SQ     |
|-------|--------|--------|
| Ent Det | SQ+WQ  | 87.6%  | 94.1%  |
| Rel Pred | SQ+WQ | 24.6%  | 81.6%  |

Table 2: Impact of training data on model effectiveness.
In Table 2, we explored the impact of training data, by applying a SQ-trained model to the validation set of WQ (and vice versa). This causes a compatibility issue for relation prediction, since there might be relations in the test set that the model did not train on. In fact, there are 271 classes in the validation set of WQ, for which there are no training instances in SQ (and 622 such classes vice versa). We report accuracy on only compatible test questions in parentheses. Additionally, we trained a combined model using all training examples in SQ and WQ (i.e., SQ+WQ), and tested on the individual datasets. For this case, since we take a union of the possible relations in SQ and WQ, there is no compatibility issue. SQ+WQ was superior in all runs.

End-to-End QA. Given a test set for the end-to-end evaluation, we apply the relation prediction and entity detection models on each test question, yielding a structured query \( q = \{e_t, r\} \) (Section 3a). We then run our search algorithm to find a ranked list of relevant entity nodes in Freebase (\( \mathcal{e}_{\text{cand}} \)), and corresponding list of answers \( \mathcal{A} \) (Section 3b). Simply comparing the top answer to the ground truth answer does not always work because our system outputs the identifier of a Freebase node, whereas ground truth answer is provided as the textual representation of a Freebase node. As a remedy, we follow Bordes et al. (2015), where we compare the predicted entity-relation pair to the ground truth. In other words, a question is counted correct if and only if the entity we find (i.e., \( \mathcal{e}_{\text{cand}} \)) matches the ground truth, and the relation we predict (i.e., \( r \)) is in the list of correct relation types (for SQ, there is only one correct relation type per question).

Table 3 summarizes end-to-end experimental results on both benchmark datasets. The upper portion shows best reported results; the lower portion is dedicated to our approach: First two columns specify which models were used to convert the question into a structured query, and third column specifies how we performed search to retrieve answers. “Best in-domain” refers to the best model that was trained on the training portion of the same corpus (Figures 1(a) and 1(b)). In order to quantify the importance of search, the first two rows show accuracy with and without search (i.e., latter case directly compares the predicted entity-relation pair in \( q, (e_t, r) \) to ground truth). Second and third rows show the effect of training models on the combined dataset instead of restricting to the training portion. As expected from Table 2, this improves the accuracy for WQ substantially (13.5% absolute points). On SQ, we notice a slight drop.

In remaining rows, a star indicates that we used the best model. The huge jump in accuracy (≈9-13%) is due to relation correction during search: For a predicted entity node \( \mathcal{e}_{\text{cand}} \), we can limit our relation choices to the set of unique relation types that \( \mathcal{e}_{\text{cand}} \) is associated with. This helps eliminate the artificial ambiguity due to many overlapping relation types in Freebase. With this simple but effective idea, the number of relation types that we need to consider per question (on average) drops from 2290 to 120 for the SQ+WQ model.

In certain cases, our approach produces the correct textual answer, although it does not pass the correctness test described above. This happens especially in SQ when the ground truth includes a single correct relation type, although there are multiple different ways to arrive at the same answer. The fourth column indicates whether these bonus cases are counted as correct answers.

In terms of comparison to prior work, our approach outperforms the state of the art by significant margins. The absolute improvement in accuracy (i.e., precision at top 1) over the state of the art is 7.1 points (15.7% relative) on WQ and 24.4 points (38.2% relative) on SQ — we obtain 52.2% and 88.3% accuracy on these two tasks, respectively.

Last three rows quantify the impact of our two RNN models by performing an ablation study, in which we replace either model with a naive baseline: (i) we assign the relation that appears most frequently in training data (i.e., born_on), and/or (ii) we tag the entire question as an entity (and then perform the n-gram based search). Results confirm that relation prediction is absolutely critical, since both datasets include a diverse and well-balanced set of relation types. When we applied the naive entity detection baseline (row 6), our results drop significantly, but they are still comparable to the state of the art. Given that most prior work actually use a
Prior work

| Author(s) | Description | Rel-pred | Entity-det | Search? | Bonus? | WQ | SQ |
|-----------|-------------|----------|------------|---------|--------|----|----|
| Berant et al., 2013 | Lambda calculus based semantic parser | | | | | 31.4 | - |
| Yang et al., 2014 | Entity embedding, uses Wikipedia | | | | | 41.3 | - |
| Reddy et al., 2014 | CCG-based semantic parser, uses ClueWeb | | | | | 41.0 | - |
| Bordes et al., 2015 | Ensemble method, uses Paraphrase DB | | | | | 42.2 | 63.9 |
| Dong et al., 2015 | Assumes entity given, CNN over Q-A pairs | | | | | 45.1 | - |

This work

| Approach | Rel-pred | Entity-det | Search? | Bonus? | WQ | SQ |
|----------|----------|------------|---------|--------|----|----|
| Best in-domain | Best in-domain | N | N | 24.6 | 71.0 |
| SQ+WQ | SQ+WQ | Y | N | 38.4 | 77.4 |
| + relation correction | | | N | 51.8 | 86.8 |
| Naive Baseline | * | * | Y | 52.2 | 88.3 |
| Most frequent | * | | Y | 2.6 | 4.1 |
| Most frequent | question is entity | | Y | 1.7 | 3.7 |

Table 3: End-to-end QA accuracy on WebQuestions (WQ) and SimpleQuestions (SQ). Boldface indicates best results in each part. Percentage in parentheses shows relative improvement over the state of the art.

variety of that baseline, this supports the argument that our RNN-based entity detection was the largest contributor to the success of our approach.

Error Analysis. In order to better understand the weaknesses of our approach, we performed a blame analysis: Among 875 errors in WQ test set, 135 can be blamed on entity detection — the relation type was correctly predicted, but the detected entity did not match the ground truth. The reverse is true for 408 cases.7 We manually labeled a sample of 50 instances from each blame scenario. When entity detection is to blame, 20% was due to spelling inconsistencies between question and KB, which can be resolved with better text normalization during indexing (e.g., “La kings” refers to “Los Angeles Kings”). We found 16% of the detected entities to be correct, even though it was not the same as the ground truth (e.g., either “New York” or “New York City” is correct in “what can do in new york?”); 18% are inherently ambiguous and need clarification (e.g., “where bin laden got killed?” might mean “Osama” or “Salem”). When blame is on relation prediction, we found that the predicted relation is correct (albeit different than ground truth) 29% of the time (e.g., “what was nikola tesla known for” can be classified as profession or notable_for). We also labeled 22% of the questions as higher-order, since they require reasoning beyond a single fact in the KB (e.g., “what country did germany invade first in ww1”).

6 Conclusions and Future work

We described a simple yet effective approach for factoid question answering. We assume that first-order QA can be reduced to two tasks, entity detection and relation prediction, and provide solutions to each with a recurrent neural network. Experimental results indicate impressive improvements over the state of the art on two commonly-used QA benchmark datasets: relative improvements in accuracy were 15.7% for WQ and 38.2% for SQ. From these very positive results, we can deduce that RNNs are very effective at tagging and classification over arbitrary text sequences. We can also argue that building a specific architecture that custom-fits the task has advantages over architectures aiming to do more general-purpose reasoning over text (e.g., (Bordes et al., 2015; Kumar et al., 2015)). Our approach would require a non-trivial augmentation to work with other types of QA tasks: “What movie did both Tom Hanks and Meg Ryan appear in?” requires us to find an entity in the KB that is related to the two entities in the question. Finally, we found that many mistakes are due to search-related issues, so a more refined retrieval could improve accuracy.

---

7 In remaining 332 incorrect answers, both models fail, so the blame is shared.
References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA. A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1533–1544. ACL.

Matthew W Bilotti, Paul Ogilvie, Jamie Callan, and Eric Nyberg. 2007. Structured Retrieval for Question Answering. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’07, pages 351–358, New York, NY, USA. ACM.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. arXiv preprint arXiv:1406.3676.

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

Davide Buscaldi, Paolo Rosso, José Manuel Gómez-Soriano, and Emilio Sanchis. 2010. Answering Questions with an N-gram Based Passage Retrieval Engine. J. Intell. Inf. Syst., 34(2):113–134, April.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In ACL (1), pages 423–433. Citeseer.

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

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Junyoung Chung, Caglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. J. Mach. Learn. Res., 12:2493–2537, November.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, pages 260–269. The Association for Computer Linguistics.

Jeffrey L Elman. 1990. Finding structure in time. Cognitive science, 14(2):179–211.

Alex Graves. 2008. Supervised sequence labelling with recurrent neural networks. Ph.D. thesis, Technical University Munich.

James Hammerton. 2003. Named entity recognition with long short-term memory. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, pages 172–175. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multiperspective sentence similarity modeling with convolutional neural networks. In Lluís Márquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, pages 1576–1586. The Association for Computational Linguistics.

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

Wei-Ning Hsu, Yu Zhang, and James R. Glass. 2016. Recurrent neural network encoder with attention for community question answering. CoRR, abs/1603.07044.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In EMNLP, pages 720–728.

Mohit Iyyer, Jordan L. Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé. 2014. A neural network for factoid question answering over paragraphs. In EMNLP.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan
Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285.

Zefu Lu, Lei Li, and Wei Xu. 2015. Twisted recurrent network for named entity recognition. In *Bay Area Machine Learning Symposium*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2786–2792. AAAI Press.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *TACL*, 2:377–392.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168*.

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

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *EMNLP*, pages 645–650.

Dell Zhang and Wee Sun Lee. 2003. Question Classification Using Support Vector Machines. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’03, pages 26–32, New York, NY, USA. ACM.