Incorporating Parsing Features for Document Reading

Ren Yi and **Dima Taji**
New York University
New York, NY
{ren.yi,dima.taji}@nyu.edu

Abstract

Question Answering (QA) is one of the more attractive problems in Natural Language Processing (NLP). In Information Retrieval (IR)-based QA system, the task of machine reading at scales tackles the challenges of document retrieval and machine comprehension of texts. In this paper, we present an approach for machine comprehension by leveraging syntactic parsing features in questions and contexts. We show that parse relation feature slightly improves performance of the current state-of-the-art QA system.

1 Introduction

Question Answering (QA) was introduced as a computational program in the 1960s (Simmons et al., 1964). The appeal of being able to use computers to answer "natural English question" was already clear, as were the challenges that this task introduced.

Several approaches have been taken to classifying the QA problem. Simmons et al. (1964) divided questions into two categories; complete and incomplete questions. Complete questions are questions that require a "yes", "no", or "to a certain extent" answer. They are classified as complete because the question itself has all the information needed, and the answer is needed to determine whether the question statement is true or false. Incomplete questions are questions that have a question word, and therefore requires the answer to introduce new information. Simmons et al. (1964) claims that the question word itself, such as "who", "what", "where", and "when", lends itself to knowing what kind of answer we are looking for. This information can be used to help in finding potential answers, and evaluating these answers to find the best possible answer.

A different approach to categorizing questions is presented by Jurafsky and Martin (2014) based on the way the answers are retrieved. They classified QA into two paradigms; information retrieval (IR) based, and knowledge based. IR based QA is when a system tries to answer a question by looking for the answer on the web, or from a set of given documents. Knowledge based QA focuses on generating a semantic representation of the question and "querying databases of facts" (Jurafsky and Martin, 2014).

Recent work has been done to improve on answering questions via IR, especially with the introduction of deep learning. This problem has several components that have been the focus of several researches; question classification (Zhang and Lee, 2003; Hermjakob, 2001; Wang et al., 2018), document retrieval (Chen et al., 2017; Tellex et al., 2003), and document reading (Chen et al., 2017; Wang et al., 2018; Hirschman and Gaizauskas, 2001).

In this paper we attempt to improve the performance of existing neural network document readers in an IR based QA system. We chose IR based systems because this is the model that is closest to human needs in QA systems. Knowledge based systems can retrieve answers from a predefined set of documents or resources, while IR based systems can be set to use the entirety of the web to answer questions. We noticed that systems suffered from a drop in performance when they tried to retrieve answers from multiple paragraphs versus a single paragraph (Chen et al., 2017). We include syntactic parsing of the answer paragraphs, since it may include an additional layer to gaging the semantics of the question

that is not represented by features such as part-of-speech (POS) tags.

In Section 2 of this paper we present some of the most significant QA systems to date. Section 3 details the approach we will be taking to attempt to improve on the existing system. Finally, Section 4 covers the data we are using, and the setup of our experiment. Section 5 illustrates the results of our experiments. A qualitative error analysis and conclusion are presented in Sections 6 and 7 respectively.

2 Related Work

This section presents some of the most recent and state-of-the-art systems that focus on IR-based question answering.

2.1 DrQA

DrQA is a question answering systems for factoid questions from Wikipedia articles (Chen et al., 2017). This system is composed of a document retriever and a document reader. The document retriever fetches the most relevant articles from Wikipedia using bi-gram hashing and term frequency inverse document frequency (TF-IDF) matching. The document reader finds the possible answer spans in these documents using a multi-layer long short-term memory network (LSTM) machine comprehension model.

The performance of the document retriever ranges between 70% and 86% where the relevant document appears in the top five documents retrieved, depending on the dataset it is tested on, which outperforms Wikipedia's search engine. The document reader performs relatively well on datasets where it is presented with one paragraph only, but when the more paragraphs are introduced - an entire Wikipedia page instead of one paragraph, the performance drops from 70% to 27%. This shows that there is a large potential for improving on the performance of this system.

2.2 R^3: Reinforced Ranker-Reader

R^3 (Wang et al., 2018) presents an improvement on the reader algorithms, by introducing a ranker that ranks the selected passage in order of likeliness of the passage containing the right answer, before passing them on to the reader. The reader is trained to predict the span of the answer similarly to the reader in DrQA (Chen et al., 2017). The ranker ranks paragraphs by comparing the question to each passage and quantifies how well the passages matches the question. In their paper, Wang et al. (2018) use passages each of the length of one sentence.

The novelty of the approach, however, stems from combining the training of the ranker and the reader using reinforcement learning. The use of the ranker improves the document retrieval part of the system from a baseline of getting the correct answer in the top 1 paragraph in 20% of the cases to 40%. It is also able to improve the reader's performance from DrQA's 28% to about 30% on SQuAD dataset (Rajpurkar et al., 2016b).

2.3 Evidence Aggregation for Answer Re-ranking

This approach was taken by Wang et al. (2017) to improve solely on the ranking component of QA systems. The philosophy behind this re-ranking approach is that an answer is a stronger candidate if it appears in multiple paragraphs. The approach they follow consists in aggregating all the passages that contained a given possible answer to form a "union passage", and measuring how much this hybrid passage matches the question.

When compared to the performance of the R^3 system (Wang et al., 2018), this approach got the answer in the top 1 paragraph in 42% of the time, which boosts the performance of the QA system to 42.3% on SQuAD.

2.4 Ask Me Anything

Ask Me Anything (Kumar et al., 2016) is a system that uses the dynamic memory network (DMN) to process a series of input statements, apply reasoning models, and answer questions related to those statements. The way this is different from the other systems we are reviewing here is that all the statements that are considered as context are connected to each other and contain information that needs some inference or reasoning. On the other hand, the other systems we discuss here contain one or more paragraphs, that may contain the same piece of information in multiple forms, therefore having no need for inference and reasoning.

This systems can also be used for sentiment analysis and POS tagging, in addition to traditional question answering. They report an accuracy of 88% on sentiment analysis, 93.6% on various question answering tasks, and 97.56% on POS tagging.

2.5 Introducing Grammar into Parsing

An attempt at using parsing to improve the performance of QA system appears in a work on using soft alignments in discriminative training (Wang et al., 2007). In this paper, quasi-synchronous grammar (QG) is used to model a tree over the question and the contexts, and an alignment is made between these trees. QG is usually used in machine translation between source and target sentences, and in this work, an answer is considered as a "free translation" of the question.

Wang et al. (2007) add a labeled, directed dependency trees, meaning that both parent and relation were modeled in the dependency tree. They look at different kinds of alignments between the question and contexts, including parent-child alignment in the question in respect to the parent-child alignment in the answer, as well as to the child-parent and grandparent-child alignments in the answer, among other kinds of alignments.

This model is evaluated on the Text REtrieval Conference (TREC) 813 QA dataset. Compared to the baseline, it drastically improves the performance: the mean average precision score improves from 51.9% to 68.1%, and the mean reciprocal rank score improves from 57.8% to 76.4%. This shows that parsing information has the potential of improving the performance of existing QA systems.

We choose to improve on the DrQA system because the system's code is readily available, and well documented. The system already performs decently on answers from one-paragraph contexts, which indicates that there is likely a good possibility to improve on its performance when faced with multiple-paragraph contexts.

3 Approach

We propose here a machine comprehension system, a document reader for extracting answers given multiple paragraphs retrieved by a standard search engine or document retriever.

Given a question $\mathbf{q} = (q_1, ..., q_{l_q})$ consisting of l_q words, and context documents consisting of l_c words from various number of paragraphs, we develop an LSTM model that learns answer locations similar to DrQA (Chen et al., 2017). Our method works as follows:

3.1 Question and context encodings

Question encoding follows the descriptions in Chen et al. (2017).

As for contexts, we represent all words w_t in the context document as a sequence of feature vectors $\mathbf{x_t}$ consisting of word embeddings, exact match, token features, aligned question embedding, and dependency parse:

- Word embeddings $f_{emb}(w_t)$, exact match $f_{match}(w_t)$, token features $f_{token}(w_t)$ and aligned question embedding $f_{align}(w_t)$ follow the descriptions in Chen et al. (2017)

- Dependency parse $f_{parse}(w_t)$: we will use the Stanford Parser (Chen and Manning, 2014) to generate dependency parse for each word:

$$f_{parse}(w_t) = [r(w_t), p(w_t)] \tag{1}$$

where $r(w_t)$ denotes the parse relation tag associated with w_t and its parent, and $p(w_t)$ denotes 300-dimensional GloVe (Pennington et al., 2014) embeddings of the parent.

Feature vector $\mathbf{x_t}$ is hence

$$\mathbf{x_t} = \begin{bmatrix} f_{emb}(w_t) \\ f_{match}(w_t) \\ f_{token}(w_t) \\ f_{align}(w_t) \\ f_{parse}(w_t) \end{bmatrix} \tag{2}$$

3.2 Model architecture.

We use the feature vectors $\mathbf{x_t}$ as input to train a multi-layer bidirectional LSTM model (bi-LSTM). The goal is to predict the span of words that most likely contains the correct answer. Specifically, for all words $\mathbf{x_t}$ we train two additional classifiers that take the output of the bi-LSTM $\mathbf{h_t}$ and the question $\mathbf{q}$, and compute its start and stop probabilities p^t_{start} and p^t_{end} as follows:

$$p^t_{start} = softmax(\mathbf{h_t W_s q}) \tag{3}$$
$$p^t_{end} = softmax(\mathbf{h_t W_e q}) \tag{4}$$

We choose the span $[i, j]$ such that $i < j < i + l$ and $p^i_{start} \times p^j_{end}$ is maximized. l is a hyperparameter of the model.

3.3 Code availability

Our method is implemented in PyTorch (Paszke et al., 2017). The software is available at `https://github.com/ryi06/MultiQA`

4 Experiment

This section describes the initial experiments setup and the dataset we will be using.

4.1 Experimental setup

We use both SQuAD (Rajpurkar et al., 2016a) and QUASAR-T (Dhingra et al., 2017) datasets for training our model. We use DrQA (Chen et al., 2017) as our baseline method to compare model performance.

4.2 Data

Both SQuAD (Rajpurkar et al., 2016a) and QUASAR-T (Dhingra et al., 2017) datasets are available in json format, but follow different structures.

DrQA is designed to work with SQuAD data structure. For this, we design a converter that converts the QUASAR-T from its current structure into that of SQuAD.

SQuAD contains 536 articles, with 23,215 paragraphs in total. It is split into 80% train, and 10% for each of dev and test. For each paragraph, SQuAD has up to five questions, with one answer for each question.

QUASAR-T is split into train, dev, and test sets, with 37,012 questions in the train set, and 3,000 questions in each of the dev and train sets. Each question has one answer to it. It also contains two different types of contexts, a long context, for which each paragraph is 15 sentences long on average, and a short context, for which each paragraph is one sentence exactly. In long contexts each question corresponds to 20 context paragraphs, and in short contexts each question corresponds to 100 context paragraphs. Not all contexts, however, may contain the answer to the question.

5 Results

This section describes the results for DrQA + relation performance on QUASAR-T and SQuAD datasets.

Paragraph	DrQA	DrQA + relation
1	84.58 ± 0.44	84.84 ± 0.43
2	79.68 ± 0.52	79.80 ± 0.31
3	76.08 ± 0.51	76.22 ± 0.47
4	74.12 ± 0.28	74.54 ± 1.02
5	71.56 ± 0.22	71.59 ± 0.44
6	70.43 ± 0.57	70.25 ± 0.29
7	68.47 ± 0.47	68.65 ± 1.32

Table 1: Comparing DrQA performance on QUASAR-T before and after adding parsing relation feature.

5.1 Evaluation on QUASAR-T

We first evaluate the performance of DrQA on QUASAR-T dataset using optimal configurations described in Chen et al. (2017). As illustrated in Figure 1, DrQA performance drops 28% as short contexts increases from 1 to 90 paragraphs, and 10% as long contexts increases from 1 to 9 paragraphs This is consistent with previous finding from (Chen et al., 2017) that DrQA performance decreases given longer context paragraphs.

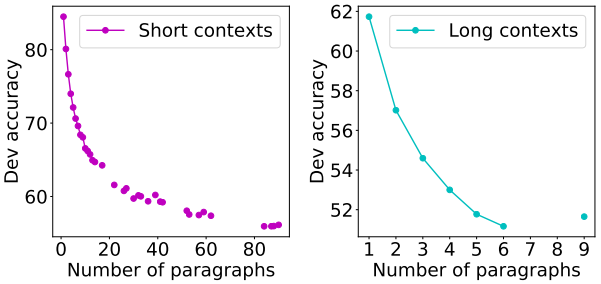

Figure 1: DrQA performance on QUASAR-T dataset as the context length increases.

Figure 2 shows QUASAR-T results when adding parse features to DrQA. All model performances decrease as the number of context paragraphs increases. Additionally, minor improvements are observed in 1-5 and 7 paragraph contexts when adding parse relation feature (Figure 2, Table 1). The most significant improvement reaches about 0.42% in 4 paragraph context. However, parse parent embedding feature reduces model performance across all paragraph contexts. It also eliminates the minor improvement from parse relation feature when both features are added to the model (DrQA + parse in Figure 2), rendering the similar level of performance decrease in DrQA + parse model.

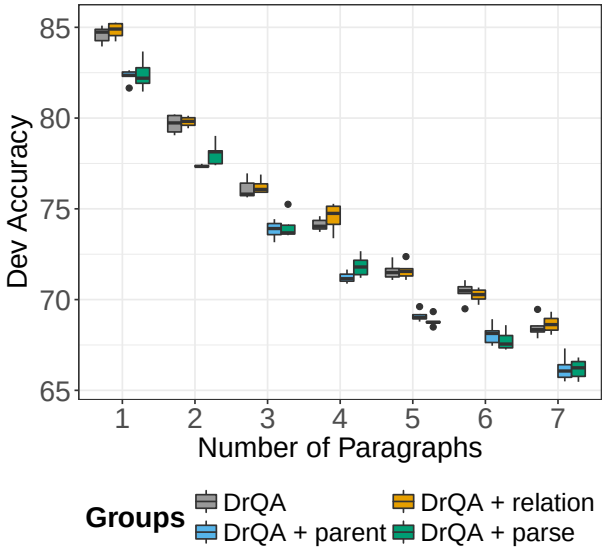

Figure 2: Model performances on QUASAR-T short context dataset with different number of paragraphs. Accuracy statistics were generated from 6 replicate experiment with different seeds.

5.2 Evaluations on SQuAD

We also access models performances on SQuAD dataset and the result is summarized in Figure 3. An average of 0.11% performance increase is observed by adding parse relation feature. However, parse parent feature decreases accuracy for about 1% compared to baseline DrQA. Although parse relation induces minor improvement compared to the baseline, adding back parse relation on top of DrQA + parent induces another 0.2% decrease. This result is roughly consistent with what we observed from QUASAR-T dataset.

6 Error Analysis

For our error analysis, we look at the prediction output of the system on the dev datasets[1]. In this section, we discuss some of the interesting examples that come up in the analysis, and we reflect on why we think the system preforms that way. The script that we are using to produce and evaluate the prediction is the same script that is part of the DrQA's code.

Example where Parsing Features Improved the Performance

[1]For space conservation, we will only show the relevant sentence in the context.

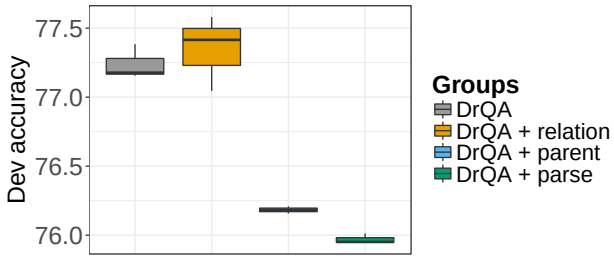

Figure 3: Model performances on SQuAD dataset. Group accuracy statistics were generated from 6 replicate experiment with different seeds. DrQA: 77.24 $\pm$ 0.13; DrQA + relation: 77.35 $\pm$ 0.27; DrQA + parent: 76.18 $\pm$ 0.04; DrQA + parse: 75.97 $\pm$ 0.04

Question: Name The Youthful Singer Who Headed The Teenagers

Context: Little Anthony's dramatic interpretation was certainly helped in the public eye by his youthful-sounding voice and name, which recalled the recently popular Frankie Lymon.

Expected Answer: Frankie Lymon

DrQA's Predicted Answer: Little Anthony with 76% probability

DrQA + relation's Predicted Answer: Frankie Lymon with 56% probability

In this particular example, the answer is not stated clearly in the provided context. We theorize that the baseline system learns that the first named entity appearing in the sentence is more likely to be the answer that it should be looking for. However, we expect that DrQA + relation's model learns how to predict the answer from parsing the context and producing the correct structure of the context and the question.

Example where Parsing Features Improved the Prediction

Question: Who Was The 1st wife of Henry VIII?

Context: to " How did Catherine of Aragon Henry VIII 's 1st wife die ?

Expected Answer: Catherine of Aragon

DrQA's Predicted Answer: Aragon with 31% probability

DrQA + relation's Predicted Answer: Catherine of Aragon with 46% probability

This is an example where the given context is not a clear grammatical sentence, however the answer is stated clearly in the sentence. We theorize that the baseline system get a part of the expected answer only because the named entity tagger does not recognize "Catherine of Aragon" as one entity. However, the dependency structure would have produced a structure where "Aragon" would be a nominal modifier of "Catherine", and "of" would be the case of "Aragon", making the phrase a subtree on its own. This would allow the system to realize that "Catherine of Aragon" all together is one entity, and therefore it's the answer to the question.

Example where Parsing Features Did Not Improved the Prediction

Question: What eats 14 feet of earthworms every day?

Context: Baby robins eat 14 feet of earthworms every day!

Expected Answer: Baby robins

DrQA's Predicted Answer: Baby robins with 50% probability

DrQA + relation's Predicted Answer: robins with 49% probability

This is an example where even though the answer produced by our system can be considered technically correct, the evaluation script using exact match would consider it a wrong answer. The parser links "Baby" to "robins" with a compound relation, which leads to the expectation that the system would treat them as one entity. However, since compound is not one of the most frequently used relations, we can also expect that the training data does not contain enough compound entities appearing as the model answer, hence it does not learn the semantic meaning of this relation.

7 Conclusion

In the above sections we analyze the effect of adding parse features on machine-comprehension system DrQA. We observe that parse relation induces minor performance improvement in SQuAD and the majority of QUASAR-T short context data tested. This is because, unlike token features, including POS tag and named entity tag, parse relation introduces non-redundant features, like nominal phrases ("Catherine of Aragon" in Section 6), that is useful for identifying answer phrases from the context. However, parse parent embedding feature reduces model performance in both datasets. We reason that parent word embedding may be redundant when word embedding is already included in the model. On the other hand, we theorize that this happens because the training datasets we used in our evaluation were all considered to be of the 'short context' category. The expected answer in most cases we saw is either the *subject* or the *object* of the main verb, and in short contexts there is usually one verb. Therefore, the parent feature does not have an affect on predicting the answer, because what matters is the syntactic role the words play. However, we expect that to change when training on longer contexts. We anticipate that in longer contexts there will be several verbs, with as many subjects and objects. Selecting the appropriate word will become more dependent on its parent, since the system will have to determine which verb it is interested in.

From the error analysis carried out on the prediction output of our system, we check the entries for which our system gives a different prediction from that of the baseline system. We found that in 41.4% of the cases, our system produces the correct answer, in 24.3% of the cases the baseline system produces the correct answer, and in 34.3% of the cases neither systems produces the correct answer.

From the qualitative analysis of these answers, we conclude that some of the cases where the prediction of our system is considered wrong are cases where we produce extra determiners, i.e. *'a'*, *'an'*, or *'the'*. We believe this is due to some inconsistency in the training data, where some answers have determiners, while others do not. This inconsistency is expected in datasets like these, however, the evaluation should take that into consideration, and not penalize for an added or a missing determiner.

Team Contribution

R.Y. and D.T. conceived the idea and designed the model. R.Y. implements the model and performs the experiments on DrQA. D.T. prepares the QUASAR-T dataset and incorporates Stanford Parser into DrQA. R.Y. and D.T. write the manuscript.

References

[Chen and Manning2014] Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

[Chen et al.2017] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

[Dhingra et al.2017] B. Dhingra, K. Mazaitis, and W. W. Cohen. 2017. Quasar: Datasets for Question Answering by Search and Reading. *ArXiv e-prints*, July.

[Hermjakob2001] Ulf Hermjakob. 2001. Parsing and question classification for question answering. In *Proceedings of the workshop on Open-domain question answering-Volume 12*, pages 1–6. Association for Computational Linguistics.

[Hirschman and Gaizauskas2001] Lynette Hirschman and Robert Gaizauskas. 2001. Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300.

[Jurafsky and Martin2014] Dan Jurafsky and James H Martin. 2014. *Speech and language processing*, volume 3. Pearson London:.

[Kumar et al.2016] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

[Paszke et al.2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

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

[Rajpurkar et al.2016a] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016a. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *ArXiv e-prints*, June.

[Rajpurkar et al.2016b] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016b. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

[Simmons et al.1964] Robert F Simmons, Sheldon Klein, and Keren McConlogue. 1964. Indexing and dependency logic for answering english questions. *Journal of the Association for Information Science and Technology*, 15(3):196–204.

[Tellex et al.2003] Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47. ACM.

[Wang et al.2007] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

[Wang et al.2017] Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2017. Evidence aggregation for answer re-ranking in open-domain question answering. *arXiv preprint arXiv:1711.05116*.

[Wang et al.2018] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced ranker-reader for open-domain question answering.

[Zhang and Lee2003] 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 informaion retrieval*, pages 26–32. ACM.