ASK THE RIGHT QUESTIONS: ACTIVE QUESTION REFORMULATION WITH REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We frame Question Answering as a Reinforcement Learning task, an approach that we call *Active Question Answering*. We propose an agent that sits between the user and a black box question-answering system and which learns to reformulate questions to elicit the best possible answers. The agent probes the system with, potentially many, natural language reformulations of an initial question and aggregates the returned evidence to yield the best answer. The reformulation system is trained end-to-end to maximize answer quality using policy gradient. We evaluate on SearchQA, a dataset of complex questions extracted from *Jeopardy!*. Our agent improves F1 by 11.4% over a state-of-the-art base model that uses the original question/answer pairs. Based on a qualitative analysis of the language that the agent has learned while interacting with the question answering system, we propose that the agent has discovered basic information retrieval techniques such as term reweighting and stemming.

1 INTRODUCTION

Web and social media have become primary sources of information. Users' expectations and information seeking activities co-evolve with the increasing sophistication of these resources. Beyond navigation, document retrieval, and simple factual question answering, users seek direct answers to complex and compositional questions. Such search sessions may require multiple iterations, critical assessment and synthesis (Marchionini, 2006).

The productivity of natural language yields a myriad of ways to formulate a question. In the face of complex information needs, humans overcome uncertainty by reformulating questions, issuing multiple searches, and aggregating responses. Inspired by humans' ability to *ask the right questions*, we present an agent that learns to carry out this process for the user. The agent sits between the user and a backend QA system that we refer to as the 'environment'. We call the agent AQA, as it implements an *active question answering* strategy. AQA aims to maximize the chance of getting the correct answer by send a reformulated question to the environment. The agent seeks to find the best answer by asking many questions and aggregating the returned evidence. The internals of the environment are not available to the agent, so it must learn to probe a black-box optimally using only question strings. The key component of our solution, Figure 1, is a sequence-to-sequence model trained with reinforcement learning (RL) using a reward based on the answer returned by the environment. The second component to AQA combines the evidence from interacting with the environment using a convolutional neural network.

We evaluate on a dataset of *Jeopardy!* questions, SearchQA (Dunn et al., 2017). These questions are hard to answer by design because they use convoluted language, e.g., *Travel doesn't seem to be an issue for this sorcerer & onetime surgeon; astral projection & teleportation are no prob* (answer: *Doctor Strange*). Thus SearchQA tests the ability of AQA to reformulate questions such that the QA system has the best chance of returning the correct answer. AQA improves over the performance of a deep network built for QA, BiDAF (Seo et al., 2017a), which has produced state-of-the-art results on multiple tasks, by 11.4% absolute F1, a 32% relative F1 improvement. We also perform a qualitative analysis of the kind of reformulations generated by the AQA agent. We hypothesize that while optimizing its output, in the context of machine-machine communication with the QA system, AQA discovers classic IR query operations such as term re-weighting, expansion and morphological

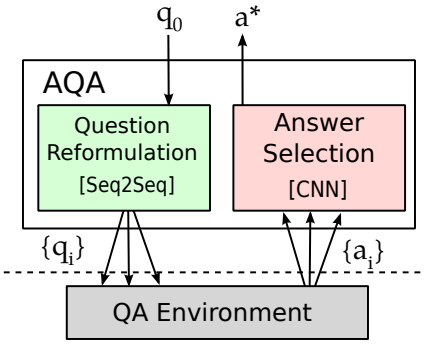

Figure 1: The AQA agent-environment setup. In the downward pass the agent reformulates the question, sending many variants to the QA system. In the upward pass the final answer is selected from several environment responses.

simplification/stemming. A possible reason being that current machine comprehension tasks involve short textual snippets ranking, thus incentivizing relevance, more than deep language understanding.[1]

2 Active Question Answering Model

Figure 1 shows the components of the Active Question Answering (AQA) agent-environment setup. The AQA model interacts with a black-box environment. AQA queries it with many versions of a question, and finally returning the best of the answers found. An episode starts with an original question q_0. The agent then generates a set of reformulations $\{q_i\}_{i=1}^N$. These are sent to the environment which returns a candidate answers $\{a_i\}_{i=1}^N$. The selection model then selects the best from these.

2.1 Question-Answering Environment

For the QA environment we use competitive neural question answering model, BiDirectional Attention Flow (BiDAF) (Seo et al., 2017a). BiDAF is an extractive QA system, it finds its answers in contiguous spans of a given document. Given a question, the environment returns an answer and, during training, a reward. The reward may be any quality metric for the returned answer, we use token-level F1 score. Note that the reward for each answer a_i is computed against the original question q_0. We assume that the environment is opaque; the agent has no access to its parameters, activations or gradients. We choose this scenario since it enables us to design a general framework that can, in principle, interact with any information source. However, without propagating gradients through the environment we lose information, feedback on the quality of the question reformulations is noisy, presenting a training challenge.

2.2 Reformulation Model

The reformulator is a sequence-to-sequence model, as is popular for neural machine translation. We build upon the implementation of Britz et al. (2017). The architecture consists of a multi-layer bidirectional LSTM, with attention-based decoding. The major departure from the standard MT setting is that our model reformulates utterances in the same language. Unlike in MT, there is little high quality training data available for monolingual paraphrasing. Effective training of highly-parametrized neural networks relies on an abundance of data. We address this challenge by first training on a related task then using signals produced during the interactions with environment.

2.3 Answer Selection Model

During training, we have access to the reward for the answer returned for each reformulation q_i. However, at test time we must predict the best answer a^*. The selection model selects the best answer from the set observed during the interaction $\{a_i\}_{i=1}^N$ For this we use a custom neural network. The network receives triples (q_0, q_i, a_i) as input and predicts the expected reward of a_i with respect to q_i. The architecture of this model consists of a 1D CNN followed by max-pooling to embed the questions

[1] Code to be open sourced upon publication.

and answer. The three resulting vectors are then concatenated and passed through a feed-forward network which produces the prediction.

3 TRAINING

3.1 QUESTION ANSWERING ENVIRONMENT

We train a model on the training set for the QA task at hand, see Section 4.4 for details. Afterwards, BiDAF becomes the black-box environment and its parameters are not updated further. In principle, we could train both the agent and the environment jointly to further improve performance. However, this is not our desired task: our aim is for the agent to learn to communicate using natural language with an environment over which is has no control.

3.2 POLICY GRADIENT TRAINING OF THE REFORMULATION MODEL

For a given question q_0, we want to return the best possible answer a^*, maximizing a reward $a^* = \text{argmax}_a R(a|q_0)$. Typically, R is the token level F1 score on the answer. The answer $a = f(q)$ is an unknown function of a question q, computed by the environment. The reward is computed with respect to the original question q_0 while the answer is provided for q. The question is generated according to a policy π_θ where θ are the policy's parameters $q \sim \pi_\theta(\,\cdot\,|q_0)$. The policy, in this case a sequence-to-sequence model, assigns a probability

$$\pi_\theta(q|q_0) = \prod_{t=1}^{T} p(w_t|w_1, \ldots, w_{t-1}, q_0) \tag{1}$$

to any possible question $q = w_1, \ldots, w_T$, where T is the length of q with tokens $w_t \in V$ from a fixed vocubulary V. The goal is to maximize the expected reward of the answer returned under the policy, $\mathbb{E}_{q\sim\pi_\theta(\cdot|q_0)}[R(f(q))]$. We optimize the reward directly with respect to parameters of the policy using Policy Gradient methods (Sutton & Barto, 1998). The expected reward cannot be computed in closed form, so we compute an unbiased estiamte with Monte Carlo sampling,

$$\mathbb{E}_{q\sim\pi_\theta(\cdot|q_0)}[R(f(q))] \approx \frac{1}{N}\sum_{i=1}^{N} R(f(q_i)), \quad q_i \sim \pi_\theta(\,\cdot\,|q_0) \tag{2}$$

To compute gradients for training we use REINFORCE (Williams & Peng, 1991),

$$\nabla\mathbb{E}_{q\sim\pi_\theta(\cdot|q_0)}[R(f(q))] = \mathbb{E}_{q\sim\pi_\theta(\cdot|q_0)}\nabla_\theta \log(\pi_\theta(q|q_0))R(f(q)) \tag{3}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log(\pi(q_i|q_0))R(f(q_i)), \quad q_i \sim \pi_\theta(\,\cdot\,|q_0) \tag{4}$$

This estimator is often found to have high variance, leading to unstable training (Greensmith et al., 2004). We reduce the variance by substracting the following baseline reward: $B(q_0) = \mathbb{E}_{q\sim\pi_\theta(\cdot|q_0)}[R(f(q))]$. This expectation is also computed by sampling from the policy given q_0.
We often observed collapse onto a sub-optimal deterministic policy. To address this we use entropy regularization

$$H[\pi_\theta(q|q_0)] = \sum_{t=1}^{T}\sum_{w_t\in V} \log(p_\theta(w_t|w_{<t}, q_0))\, p_\theta(w_t|w_{<t}, q_0) \tag{5}$$

This final objective is:

$$\mathbb{E}_{q\sim\pi_\theta(\cdot|q_0)}[R(f(q)) - B(q_0)] + \lambda H[\pi(q|q_0)], \tag{6}$$

where λ is the regularization weight.

3.3 ANSWER SELECTION

Unlike the reformulation policy, we train the answer With either beam search or sampling we can produce many rewrites of a single question from our reformulation system. We issue each rewrite to the QA environment, yielding a set of (query, rewrite, answer) tuples from which we need to pick the best instance. We train another neural network to pick the best answer from the candidates. We

frame the task as binary classification, distinguishing between above and below average performance. In training, we compute the F1 score of the answer for every instance. If the rewrite produces an answer with an F1 score greater than the average score of the other rewrites the instance is assigned a positive label. We ignore questions where all rewrites yield equally good/bad answers. We evaluated FFNNs, LSTMs and CNNs and found that the performance of all systems was comparable. Since the inputs are triples of variable length sequences the latter two allow us to incorporate the tokens directly without the need for feature engineering. We choose a CNN for computational efficiency. We use pre-trained embeddings for the tokens of query, rewrite, and answer. For each, we add a 1-D CNN followed by max-pooling. The three resulting vectors are then concatenated and passed through a feed-forward network which produces the binary output.

3.4 Pretraining of the Reformulation Model

We pre-train the policy by building a paraphrasing Neural MT model that can translate English to English. While parallel corpora are available for many language pairs, English-English corpora are scarce. We first produce a multilingual translation system that translates between several languages (Johnson et al., 2016). This allows us to use available bilingual corpora. Multilingual training requires nothing more than adding two special tokens to the data indicating the source and target languages. The encoder-decoder architecture of the translation model remains unchanged.

As Johnson et al. (2016) show, this model can be used for *zero-shot translation*, i.e. to translate between language pairs for which it has seen no training examples. For example after training English-Spanish, English-French, French-English, and Spanish-English the model has learned a single encoder that encodes English, Spanish, and French and a decoder for the same three languages. Thus, we can use the same model for French-Spanish, Spanish-French and also English-English translation by adding the respective tokens to the source. Johnson et al. (2016) note that zero-shot translation ususally performs worse than bridging, an approach that uses the model twice: first, to translate into a pivot language, then into the target language. However, the performance gap can be closed by running a few training steps for the desired language pair. Thus, we first train on multilingual data, then on a small corpus of monolingual data.

4 Experiments

4.1 Question Answering Data

SearchQA (Dunn et al., 2017) is a recently-released dataset built starting from a set of *Jeopardy!* clues. Clues are obfuscated queries such as *This 'Father of Our Country' didn't really chop down a cherry tree*. Each clue is associated with the correct answer, e.g. *George Washington*, and a list of snippets from Google's top search results. SearchQA contains over 140k question/answer pairs and 6.9M snippets. We train our model on the pre-defined training split, perform model selection and tuning on the validation split and report results on the validation and test splits. The training, validation and test sets contain 99,820, 13,393 and 27,248 examples, respectively.

4.2 Question Reformulator Training

For the pre-training of the reformulator we use the multilingual United Nations Parallel Corpus v1.0 (Ziemski et al., 2016). This dataset contains 11.4M sentences which are fully aligned across six UN languages: Arabic, English, Spanish, French, Russian, and Chinese. From all bilingual pairs we produce a multilingual training corpus of 30 language pairs. This yields 340M training examples which we use to train the zero-shot neural MT system (Johnson et al., 2016). We tokenize our data using 16k sentence pieces.[2] Following Britz et al. (2017) we use a bidirectional LSTM as encoder and a 4-layer stacked LSTM with attention as decoder. The model converged after training on 400M instances using the Adam optimizer with a learning rate of 0.001 and batch size of 128.

The model trained as described above has poor quality. For example, for the question *What month, day and year did Super Bowl 50 take place?*, the top rewrite is *What month and year goes back to the morning and year?*. To improve quality, we resume training on a smaller monolingual dataset, extracted from the Paralex database of question paraphrases (Fader et al., 2013).[3] Unfortunately, this data contains many noisy pairs. We filter many of these pairs out by keeping only those where the Jaccard coefficient between the sets of source and target terms is above 0.5. Further, since the

[2] `https://github.com/google/sentencepiece`

[3] `http://knowitall.cs.washington.edu/paralex/`

number of paraphrases for each question can vary significantly, we keep at most 4 paraphrases for each question. After processing, we are left with about 1.5M pairs out of the original 35M. The refined model has visibly better quality than the zero-shot one; for the example question above it generates *What year did superbowl take place?*. We also tried training on the monolingual pairs alone. As in (Johnson et al., 2016), the quality was in between the multilingual and refined models.

After pre-training the reformulator, we switch the optimizer from Adam to SGD and train for 100k RL steps of batch size 64 with a low learning rate of 0.001. We use an entropy regularization weight of $\lambda = 0.001$. For a stopping criterion, we monitor the reward from the best single rewrite, generated via greedy decoding, on the validation set. In contrast to our initial training which we ran on GPUs, this training phase is dominated by latency of the QA system and we run inference and updates on CPU and the BiDAF environment on GPU.

4.3 Training the Aggregator

For the aggregator we use supervised learning: first, we train the reformulator, then we generate $N = 20$ rewrites for each question in the SearchQA training and validation sets. After sending these to the environment we have about 2M (question, rewrite, answer) triples to train the aggregator. We remove queries where all rewrites yield identical rewards, which removes about half of the aggregation training data. We use pre-trained 100-dimensional embeddings (Pennington et al., 2014) for the tokens. Our CNN-based aggregator encodes the three strings into 100 dimensional vectors using a 1D CNN with kernel width 3 and output dimension 100 over the embedded tokens, followed by max-pooling. The vectors are then concatenated and passed through a feed-forward network which produces the binary output, indicating whether the triple performs below of above average, relative to the other reformulations and respective answers.

We use the training portion of the SearchQA data thrice, first for the initial training of the BiDAF model, then for the reinforcement-learning based tuning of the reformulator, and finally for training of the aggregator. We carefully monitored that this didn't cause severe overfitting. BiDAF alone has a generalization gap between the training and validation set errors of 3.4 F1. This gap remains virtually identical after training the rewriter. After training the CNN, AQA-Full has a slightly larger gap of 3.9 F1. We conclude that training AQA on BiDAF's training set causes very little additional overfitting. We use the test set only for evaluation of the final model.

4.4 Baselines and Benchmarks

As a baseline, we repeat the results reported for a deep learning system developed for SearchQA (Dunn et al., 2017). This is a modified pointer network, called Attention Sum Reader.[4]

The BiDAF environment can be used without the reformulator to answer the original question. This corresponds to the raw performance of BiDAF, and is our second baseline. We train BiDAF directly on the SearchQA training data. In the SearchQA task, the answers are augmented with several snippets (50 on average) returned by a Google Search for the question. We join snippets to form the context from which BiDAF selects answer spans. For performance reasons, we limit the context to the top 10 snippets. This corresponds to finding the answer on the first page of Google results. The results are only mildly affected by this limitation, for 10% of the questions there is no answer in this shorter context. These datapoints are all counted as losses. We trained with the Adam optimizer for 4500 steps, using learning rate 0.001, batch size 60.

We present two benchmark performance levels. The first is human performance reported in (Dunn et al., 2017) based on a sample of the test set. The second is 'Oracle Ranking', which provides an upper bound on the improvement that can be made by aggregation. For this, we replace the aggregator with an oracle that picks the answer with highest F1 score from the set of those returned for the reformulations.

4.5 AQA Variants

We evaluate several variants of AQA. For each query q in the evaluation we generate a list of reformulations q_i, for $i = 1 \ldots N$, from the AQA reformulator trained as described in Section 3. We set $N = 20$ in these experiments.

AQA Top Hyp. First, we omit aggregation, and just select the top hypothesis generated by the sequence model, q_1.

[4] Dunn et al. (2017) also provide a simpler baseline that ranks unigrams from the search snippets by their TF-IDF score. This baseline is not comparable to our experiments as it can only return unigram answers.

Table 1: Results table for the experiments on SearchQA.

	Validation		Test	
System	Exact Match	F1	Exact Match	F1
Att. Sum Reader	–	24.2	–	22.8
BiDAF	31.7	37.9	28.6	34.6
AQA Top Hyp.	32.0	38.2	30.6	36.8
AQA Voting	33.6	40.5	33.3	39.3
AQA Max Conf.	35.5	42.0	33.8	40.2
AQA Full	**40.8**	**47.6**	**39.0**	**46.0**
Human Performance	–	–	43.9	–
Oracle Ranking	48.3	56.0	46.6	54.3

Source	Length	TF-mean	DF-mean	DF-median
SearchQA	9.63	1.03	1590.0	457.9
Base-NMT	6.29	1.01	362.1	60.1
AQA-R	11.9	1.20	664.4	105.8

Table 2: Results of the qualitative analysis on SearchQA.

AQA Voting In addition to the most likely answer span, BiDAF also reports a model score, which we use for a heuristic weighted voting scheme to implement a deterministic aggregator. Let a be the answer returned by BiDAF for query q, with associated score $s(a)$. We pick the answer according to $\text{argmax}_a \sum_{a'=a} s(a')$.

AQA Max Conf. We implement a second heuristic aggregator that selects the answer with the single highest BiDAF score across question reformulations.

AQA Full Finally, we present the complete system with the learned CNN aggregation model described in Section 2.

4.6 Results

Table 1 shows the results. We report exact match and F1 metrics, computed on token level between the predicted answer and the gold answer. We present results on the full validation and test sets (referred to as n-*gram* in (Dunn et al., 2017)). This includes questions that have both unigram and longer answers.

SearchQA appears to be harder than other recent QA tasks such as SQuAD (Rajpurkar et al., 2016), for both machines and humans. BiDAF's performance drops by 40 F1 points on SearchQA compared to SQuAD. However, BiDAF is still competitive on SeachQA, improving over the Attention Sum Reader network by 13.7 F1 points.

Using the top hypothesis alone already yields an improvement of 2.2 F1 on test. This improvement without aggregation demonstrates that the reformulator is able to produce questions more easily answered by the environment. Heuristic aggregation via both Voting and Max Conf yield a further performance boost. Both heuristics draw upon the intuition that when BiDAF is confident in its answer it is more likely to be correct, and that multiple instances of the same answer provide positive evidence (for Max Conf, the max operation implicitly rewards having an answer scored with respect to multiple questions).

Finally, a trained aggregation function improves performance further, yielding an absolute increase of 11.4 F1 points (32% relative) over BiDAF with the original questions. In terms of exact match score this more than closes half the gap between BiDAF and human performance.

4.7 Qualitative analysis

We analyze here some of the properties of the agents question reformulations. We use the dev partition of SearchQA, which contains 12635 questions. It is important to keep in mind that the Jeopardy! clues in the SearchQA dataset are pre-processed by lower-casing and stop word removal.

The overall effect is that the pre-processed clues are closer to keyword-based search queries than grammatical questions; e.g., “Gandhi was deeply influenced by this count who wrote "War and Peace"” is simplified to “gandhi deeply influenced count wrote war peace”. Table 4.7 summarizes the results of this analysis. The SearchQA questions are 9.6 words long on average. They contain few repeated terms, computed as the mean term frequency (TF) per question. The average is 1.03 on the dev set. We also analyze the mean and median document frequency (DF) per query. This gives us a measure of how informative the question terms are.[5]

We first consider the top hypothesis generated by the pre-trained NMT system (Base-NMT). This is the starting point for the agent training. The Base-NMT rewrites look very different from the corresponding sources. They are shorter, 6.3 words on average, and they look more like syntactically well-formed questions. E.g., the example above becomes ”Who influenced count wrote war?’.[6] It is interesting to note how Base-NMT recovers missing grammatical components (function words). We verified the improved fluency also using a large language model and found that the Base-NMT rewrites are 50% more likely than the original ones.[7]. Base-NMT rewrites are much shorter than the sources (6.29 on average) and are also characterized by few repeated terms (1.01). These rewrites also involve rarer terms in terms of DF. This is probably due to the shorter sequences and the domain mismatch between SearchQA and the NMT training corpus.

We then considered the top hypothesis generated by AQA’s reformulator (AQA-R), after policy gradient training. These are the rewrites evaluated as *AQA Top Hyp.* in Table 1. We use these rewrites instead of the output of the full system to avoid confounding effects due to the CNN rescoring. The rewrites look very different from both the Base-NMT and the SearchQA ones. For the example above AQA-R’s top hypothesis is “What is name gandhi gandhi influence wrote peace peace?”. Somewhat mysteriously, 99.8% start with the prefix “What is name”.[8] This is puzzling as it happens only for 9 Base-NMT rewrites, and never in the original SearchQA questions. We speculate it might be related to the fact that virtually all answers involve names, of named entities (Micronesia) or generic concepts (pizza). AQA-R’s rewrites are visibly less fluent than both the originals and the Base-MT ones.[9]. AQA-R’s rewrites have more repeated terms (1.2 average TF), and while they are significantly longer (11.9) terms are rarer than in the SearchQA questions (both DF measures).[10] Finally, we report that AQA-R’s reformulations often, 12.5% of the time, involve *morphological variants*. The number of questions that contain multiple tokens with the same stem doubles from SearchQA to AQA-R. Singular forms are preferred over plurals. Morphological simplification is useful because it increases the chance that word variants match the same embeddings.

Recently, Lewis et al. (2017) trained chatbots that negotiate via language utterances in order to complete a task. They report that the agents’ language can diverge from human language if there is no incentive to do so in the reward function. Our findings seem related and might help shed further light on the kind of language agents can develop in machine-machine communication. Clearly, the fact that the questions reformulated by AQA bare little resemblance to natural language does not depend on the keyword-like inputs, since Base-NMT is perfectly capable of producing fluent questions from the same input. We hypothesize instead that there is no incentive for the model to do so because of the environment and ultimately due to the nature of the task. AQA learns to ask BiDAF questions by optimizing a language that increases the likelihood of BiDAF “extracting” the right answer. Jia & Liang (2017) recently argued that reading comprehension systems are not capable of significant language understanding and can be challenged easily in adversarial settings. We propose that current deep QA systems are primarily sophisticated ranking systems trained to sort short snippets of text from the context. As such they resemble document retrieval systems and incentivizes the discovery of the kind of IR techniques that have been successful for decades: term re-weighting, query expansion and stemming/morphological simplification. Interestingly, some of the techniques learned seem to adapt to the specific properties of current deep QA architectures such as character-based modeling and attention. Sometimes AQA learns to generate semantically inappropriate surface term variants because they can be exploited by the character-based encoder; e.g., in the example above “eastern” is

[5]Here a document is the context associated with each question, i.e., the search results. The median is useful to rule the effect of frequent outliers.

[6]More examples can be found in Appendix A.

[7]Normalizing for length. A t-test for equal means confirms that the difference is significant for $p < 0.00001$.

[8]The second most frequent is “What country is” (81 times), then “What is is” (70) and "What state” (14).

[9]Also confirmed by the language model check.

[10]The differences between AQA-R and SearchQA are stastically significant according to a t-test for equal means for $p < 0.00001$.

rewritten as "eastery". Further, repetitions can increment the chances of alignment in the attention components.

5 RELATED WORK

Query operations involving reformulation, expansion and term reweighting are classic topics in information retrieval (Baeza-Yates & Ribeiro-Neto, 1999, chap. 5). In the natural language processing literature, Lin & Pantel (2001) proposed an early approach to learn patterns of question variants by comparing dependency parsing trees. Later, Duboue & Chu-Carroll (2006) showed that MT-based paraphrases can be useful in principle by providing significant headroom in oracle-based estimations of QA performance. More recently, Berant & Liang (2014) used paraphrasing to augment the training of a semantic parser by expanding through the paraphrases as a latent representation. Bilingual corpora and machine translation have been used to generate paraphrases by pivoting through a second language. Recent work uses neural translation models and multiple pivots (Mallinson et al., 2017). In contrast, our approach does not use pivoting and, to our knowledge, is the first direct neural paraphrasing system. Riezler et al. (2007) propose a phrase-based paraphrasing system for query expansion. In contrast with this line of work our goal is to generate full question reformulations while optimizing directly the end-to-end target quality metrics.

Reinforcement learning is gaining traction in natural language understanding across many problems. For example, Narasimhan et al. (2015) use RL to learn control policies for multi-user dungeon games where the state of the game is summarized by a textual description, and Li et al. (2017) use RL for dialogue generation. Policy gradient methods have been investigated recently for MT and other sequence-to-sequence problems. They alleviate limitations inherent to the word-level optimization of the cross-entropy loss, allowing sequence-level reward functions, like BLEU, to be used. Sequence level reward functions based on language models and reconstruction errors are used to bootstrap MT with fewer resources (Xia et al., 2016). Bahdanau et al. (2016) extend this line of work using actor-critic training for MT. RL training can also prevent *exposure bias*; an inconsistency between training and inference time stemming from the fact that the model never sees its own mistakes during training (Ranzato et al., 2015). We also use policy gradient to optimize our agent, however, we use end-to-end question answering quality as the reward.

Uses of policy gradient for QA include Liang et al. (2017), who train a semantic parser to query a knowledge base, and Seo et al. (2017b) who propose query reduction networks that transform a query to answer questions that involve multi-hop common sense reasoning. The work of Nogueira & Cho (2016) is most related to ours. Their goal is to identify a document containing an answer to a question by following links on a document graph. Evaluating on a set of questions from the game "Jeopardy!", they learn to walk the Wikipedia graph using an RNN until they reach the predicted article/answer. In a recent follow-up Nogueira & Cho (2017) improve document retrieval with an approach inspired by relevance feedback in combination with RL. They reformulate a query by adding terms from documents retrieved from a search engine for the original query. Our work differs in that we generate full reformulations via sequence-to-sequence modeling rather than adding single terms, and we target question-answering, rather than document retrieval.

Active QA is also related to recent research on fact-checking: Wu et al. (2017) propose to perturb database queries in order to estimate the support of quantitative claims. In Active QA questions are perturbed with a similar purpose, although directly at the surface natural language form.

6 CONCLUSION

We propose a new framework to improve question answering. We call it active question answering (AQA), as it aims to improve answering by systematically perturbing input questions. We investigated a first system of this kind that has three components: a question reformulator, a black box QA system, and a candidate answer aggregator. The reformulator and aggregator form a trainable agent that seek to elicit the best answers from the QA system. Importantly, the agent may only query the environment with natural language questions. Experimental results prove that the approach is highly effective. We improve a sophisticated Deep QA system by 11% absolute F1, 32% relative F1, on a difficult dataset of long, semantically complex, questions. For future work we will investigate the sequential, iterative aspect of QA, and frame the problem as an end-to-end RL task, thus, closing the loop between the reformulator and the aggregator.

REFERENCES

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., 1999.

D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. An actor-critic algorithm for sequence prediction. *http://arxiv.org/abs/1607.07086*, 2016.

J. Berant and P. Liang. Semantic parsing via paraphrasing. In *Proceedings of ACL*, 2014.

D. Britz, A. Goldie, M. Luong, and Q. Le. Massive exploration of neural machine translation architectures. *http://arxiv.org/abs/1703.03906*, 2017.

P. A. Duboue and J. Chu-Carroll. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proceedings of HLT-NAACL*, 2006.

M. Dunn, L. Sagun, M. Higgins, U. Guney, V. Cirik, and K. Cho. SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine. *http://arxiv.org/abs/1704.05179*, 2017.

A. Fader, L. Zettlemoyer, and O. Etzioni. Paraphrase-Driven Learning for Open Question Answering. In *Proceedings of ACL*, 2013.

E. Greensmith, P. Bartlett, and J. Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5, 2004.

R. Jia and P. Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of EMNLP*, 2017.

M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *http://arxiv.org/abs/1611.04558*, 2016.

M. Lewis, D. Yarats, Y. Dauphin, D. Parikh, and D. Batra. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of EMNLP*, 2017.

J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky. Deep reinforcement learning for dialogue generation. *http://arxiv.org/abs/1606.01541*, 2017.

C. Liang, J. Berant, Q. Le, K.D. Forbus, and N. Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of ACL*, 2017.

D. Lin and P. Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4), 2001.

J. Mallinson, R. Sennrich, and M. Lapata. Paraphrasing revisited with neural machine translation. In *Proceedings of EACL*, 2017.

G. Marchionini. Exploratory search: From finding to understanding. *Commun. ACM*, 49(4), 2006.

K. Narasimhan, T. Kulkarni, and R. Barzilay. Language understanding for text-based games using deep reinforcement learning. *http://arxiv.org/abs/1506.08941*, 2015.

R. Nogueira and K. Cho. End-to-end goal-driven web navigation. In *Proceedings of NIPS*, 2016.

R. Nogueira and K. Cho. Task-oriented query reformulation with reinforcement learning. *http://arxiv.org/abs/1704.04572v1*, 2017.

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, 2014.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of EMNLP*, 2016.

M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *http://arxiv.org/abs/1511.06732*, 2015.

S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of ACL*, 2007.

M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional Attention Flow for Machine Comprehension. In *Proceedings of ICLR*, 2017a.

M. Seo, S. Min, A. Farhadi, and H. Hajishirzi. Query-reduction networks for question answering. In *Proceedings of ICLR 2017*, 2017b.

R.S. Sutton and A.G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

R.J. Williams and J. Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3), 1991.

Y. Wu, P.K. Agarwal, C. Li, J. Yang, and C. Yu. Computational fact checking through query perturbations. *ACM Transactions on Database Systems*, 42(1), 2017.

Y. Xia, D. He, T. Qin, L. Wang, N. Yu, T. Liu, and W. Ma. Dual learning for machine translation. In *Proceedings of NIPS*, 2016.

M. Ziemski, M. Junczys-Dowmunt, and B. Poliquen. The united nations parallel corpus v1.0. In *Proceedings of LREC*, 2016.

A ANALYSIS

Original	Gandhi was deeply influenced by this count who wrote "War and Peace"
SearchQA	gandhi deeply influenced count wrote war peace
Base-NMT	Who influenced count wrote war?
Agent	What is name gandhi gandhi influence wrote peace peace?
Original	Used underwater, ammonia gelatin is a waterproof type of this explosive
SearchQA	used underwater , ammonia gelatin waterproof type explosive
Base-NMT	Where is ammonia gelatin waterproof?
Agent	What is name is used under water with ammonia gelatin water waterproof type explosive?