

RankQA: Neural Question Answering with Answer Re-Ranking

Bernhard Kratzwald[♣] Anna Eigenmann[◇] Stefan Feuerriegel[♣]

[♣] Chair of Management Information Systems, ETH Zurich

[◇] Department of Mathematics, ETH Zurich

{bkratzwald, eianna, sfeuerriegel}@ethz.ch

Abstract

The conventional paradigm in neural question answering (QA) for narrative content is limited to a two-stage process: first, relevant text passages are retrieved and, subsequently, a neural network for machine comprehension extracts the likeliest answer. However, both stages are largely isolated in the status quo and, hence, information from the two phases is never properly fused. In contrast, this work proposes RankQA¹: RankQA extends the conventional two-stage process in neural QA with a third stage that performs an additional answer re-ranking. The re-ranking leverages different features that are directly extracted from the QA pipeline, i. e., a combination of retrieval and comprehension features. While our intentionally simple design allows for an efficient, data-sparse estimation, it nevertheless outperforms more complex QA systems by a significant margin: in fact, RankQA achieves state-of-the-art performance on 3 out of 4 benchmark datasets. Furthermore, its performance is especially superior in settings where the size of the corpus is dynamic. Here the answer re-ranking provides an effective remedy against the underlying noise-information trade-off due to a variable corpus size. As a consequence, RankQA represents a novel, powerful, and thus challenging baseline for future research in content-based QA.

1 Introduction

Question answering (QA) has recently experienced considerable success in variety of benchmarks due to the development of neural QA (Chen et al., 2017; Wang et al., 2018). These systems largely follow a two-stage process. First, a module for information retrieval selects text passages which appear relevant to the query from the cor-

pus. Second, a module for machine comprehension extracts the final answer, which is then returned to the user. This two-stage process is necessary for condensing the original corpus to passages and eventually answers; however, the dependence limits the extent to which information is passed on from one stage to the other.

Extensive efforts have been made to facilitate better information flow between the two stages. These works primarily address the interface between the stages (Lee et al., 2018; Lin et al., 2018), i. e., which passages and how many of them are forwarded from information retrieval to machine comprehension. For instance, the QA performance is dependent on the corpus size and the number of top- n passages that are fed into the module for machine comprehension (Kratzwald and Feuerriegel, 2018). Nevertheless, machine comprehension in this approach makes use of only limited information (e. g., it ignores the confidence or similarity information computed during retrieval).

State-of-the-art approaches for selecting better answers engineer additional features within the machine comprehension model with the implicit goal of considering information retrieval. For instance, the DrQA architecture of Chen et al. (2017) includes features pertaining to the match between question words and words in the paragraph. Certain other works also incorporate a linear combination of paragraph and answer score (Lee et al., 2018). Despite that, the use is limited to simplistic features and the potential gains of re-ranking remain untapped.

Prior literature has recently hinted at potential benefits from answer re-ranking, albeit in a different setting (Wang et al., 2017): the authors studied multi-paragraph machine comprehension at sentence level, instead of a complete QA pipeline involving an actual information retrieval module

¹Code is available from <https://github.com/bernhard2202/rankqa>

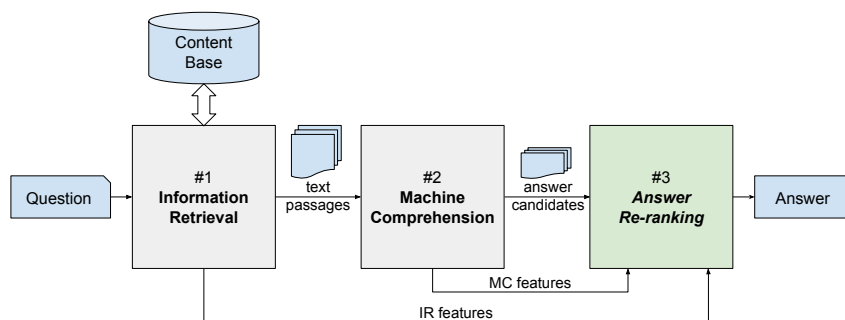


Figure 1: The RankQA system consisting of three modules for information retrieval, machine comprehension, and our novel answer re-ranking. RankQA fuses information from the information retrieval and machine comprehension phase to re-rank answer candidates within a full neural QA pipeline.

over a full corpus of documents. However, when adapting it from a multi-paragraph setting to a complete corpus, this type of approach is known to become computationally infeasible (cf. discussion in Lee et al., 2018). In contrast, answer re-ranking as part of an actual QA pipeline not been previously studied.

Proposed RankQA: This paper proposes a novel paradigm for neural QA. That is, we augment the conventional two-staged process with an additional third stage for efficient answer re-ranking. This approach, named “RankQA”, overcomes the limitations of a two-stage process in the status quo whereby both stages operate largely in isolation and where information from the two is never properly fused. In contrast, our module for answer re-ranking fuses features that stem from both retrieval and comprehension. Our approach is intentionally light-weight, which contributes to an efficient estimation, even when directly integrated into the full QA pipeline. We show the robustness of our approach by demonstrating significant performance improvements over different QA pipelines.

Contributions: To the best of our knowledge, RankQA represents the first neural QA pipeline with an additional third stage for answer re-ranking. Despite the light-weight architecture, RankQA achieves state-of-the-art performance across 3 established benchmark datasets. In fact, it even outperforms more complex approaches by a considerable margin. This particularly holds true when the corpus size is variable and where the resulting noise-information trade-off requires an effective remedy. Altogether, RankQA yields a strong new baseline for content-based question answering.

2 RankQA

RankQA is designed as a pipeline of three consecutive modules (see Fig. 1), as detailed in the following. Our main contribution lies in the design of the answer re-ranking component and its integration into the full QA pipeline. In order to demonstrate the robustness of our approach, we later experiment with two implementations in which we vary module 2.

2.1 Module 1: Information Retrieval

For a given query, the information retrieval module retrieves the top- n (here: $n = 10$) matching documents from the content repository and then splits these articles into paragraphs. These paragraphs are then passed on to the machine comprehension component. The information retrieval module is implemented analogously to the default specification of Chen et al. (2017), scoring documents by hashed bi-gram counts.

2.2 Module 2: Machine Comprehension

The machine comprehension module extracts and scores one candidate answer for every paragraph of all top- n documents. Hence, this should result in $\gg n$ candidate answers; however, out of these, the machine comprehension module selects only the top- k candidate answers $[c_1, \dots, c_k]$, which are then passed on to the re-ranker. The size k is a hyperparameter (here: $k = 40$). We choose two different implementations for the machine comprehension module in order to show the robustness of our approach.

Implementation 1 (DrQA): Our first implementation is based on the DrQA document reader (Chen et al., 2017). This is the primary system in our experiments for two reasons. First, in neu-

ral QA, DrQA is a well-established baseline. Second, DrQA has become a widespread benchmark with several adaptations, which lets us compare our approach for answer re-ranking with other extensions that improve the retrieval of paragraphs (Lee et al., 2018) or limit the information flow between the retrieval and comprehension phases (Kratzwald and Feuerriegel, 2018).

Implementation 2 (BERT-QA): QA systems whose machine comprehension module is based on BERT are gaining in popularity (Yang et al., 2019a,b). Following this, we implement a second QA pipeline where the document reader from DrQA is replaced with BERT (Devlin et al., 2019).² We call this system BERT-QA and use it as a second robustness check in our experiments.

2.3 Module 3: Answer Re-Ranking

Our re-ranking module receives the top- k candidate answers $[c_1, \dots, c_k]$ from the machine comprehension module as input. Each candidate c_i , $i = 1, \dots, k$, consists of the actual answer span s_i (i. e., the textual answer) and additional meta-information ϕ_i such as the document ID and paragraph ID from which it was extracted. Our module follows a three-step procedure in order to re-rank answers:

- (i) **Feature extraction:** First, we extract a set of information retrieval and machine comprehension features for every answer candidate directly from the individual modules of the QA pipeline.
- (ii) **Answer aggregation:** It is frequently the case that several answer candidates c_i are duplicates and, hence, such identical answers are aggregated. This creates additional aggregation features, which should be highly informative and thus aid the subsequent re-ranking.
- (iii) **Re-ranking network:** Every top- k answer candidate is re-ranked based on the features generated in (i) and (ii).

2.3.1 Feature Extraction

During this step, we extract several features from the information retrieval and machine comprehension modules for all top- k answer candidates,

²We used the official implementation from <https://github.com/google-research/bert>

which can later be fused; see a detailed overview in Tbl. 1. These features are analogously computed by most neural QA systems, albeit for other purposes than re-ranking. Nevertheless, this fact should highlight that such features can be obtained without additional costs. The actual set of features depends on the implementation of the QA system (e. g., DrQA extracts additional named entity features, as opposed to BERT-QA).

From the *information retrieval* module, we obtain: (i) the document-question similarity; (ii) the paragraph-question similarity; (iii) the paragraph length; (iv) the question length; and (v) indicator variables that specify with which word a question starts (e. g., “what”, “who”, “when”, etc.).

From the *machine comprehension* module, we extract: (i) the original score of the answer candidate; (ii) the original rank of the candidate answer; (iii) part-of-speech tags of the answer; and (iv) named entity features of the answer. The latter two are extracted only for DrQA and encoded via indicator variables that specify whether the answer span contains a named entity or part-of-speech tag (e. g., PERSON=1 or NNS=1).

2.3.2 Answer Aggregation

It is frequently the case that several candidate answers are identical and, hence, we encode this knowledge as a set of additional features. The idea of answer aggregation is similar to Lee et al. (2018) and Wang et al. (2017), although there are methodological differences: the previous authors sum the probability scores for identical answers, whereas the aim in RankQA is to generate a rich set of aggregation features.

That is, we group all answer candidates with an identical answer span. Formally, we merge two candidate answers c_i and c_j if their answer span is equal, i. e., $s_i = s_j$. We keep the information retrieval and machine comprehension features of the initially higher-ranked candidate $c_{\min\{i,j\}}$. In addition, we generate further aggregation features as follows: (i) the number of times a candidate with an equal answer span appears within the top- k candidates; (ii) the rank of its first occurrence; (iii) the sum, mean, minimum, and maximum of the span scores; and (iv) the sum, mean, minimum, and maximum of the document-question similarity scores. Altogether, this results, for each candidate answer c_i , in a vector x_i containing all features from information retrieval, machine comprehension, and answer aggregation.

Feature Group	Description	Aggregation	Impl.
INFORMATION RETRIEVAL FEATURES			
Document-query similarity	Similarity between the question and the full document the answer was extracted from.	min, max, avg, sum	both
Paragraph-query similarity	Similarity between the question and the paragraph the answer was extracted from.	—	both
Length features	Length of the document, length of the paragraph, and length of the question.	—	both
Question type	The question type is a 13-dimensional vector indicating weather the questions started with the words What was, What is, What, In what, In which, In, When, Where, Who, Why, Which, Is, or <other>.	—	both
MACHINE COMPREHENSION FEATURES			
Span features	The score of the answer candidate as assigned directly from the MC module, proportional to the probability of the answer given the paragraph, i. e., $\propto p(a p)$.	min, max, avg, sum	both
Named entity features	A 13-dimensional vector indicating whether one of following 13 named entities is contained within the answer span: location, person, organization, money, percent, date, time, set, duration, number, ordinal, misc, and <other>.	—	only 1
Part-of-speech features	A 45-dimensional vector indicating which part-of-speech tag is contained within the answer span. We use the Penn Treebank PoS tagset. (Marcus et al., 1993).	—	only 1
Ranking	Original ranking of the answer candidate.	number of occurrences	both

Table 1: Detailed description of all features used in our answer re-ranking component.

2.3.3 Re-Ranking Network

Let $x_i \in \mathbb{R}^d$ be the d -dimensional feature vector for the answer candidate c_i , $i = 1, \dots, k$. We score each candidate via the following ranking network, i. e., a two-layer feed-forward network $f(x_i)$ that is given by

$$f(x_i) = \text{ReLU}(x_i A^T + b_1) B^T + b_2, \quad (1)$$

where $A \in \mathbb{R}^{m \times d}$ and $B \in \mathbb{R}^{1 \times m}$ are trainable weight matrices and where $b_1 \in \mathbb{R}^m$ and $b_2 \in \mathbb{R}$ are linear offset vectors.

During our experiments, we tested various ranking mechanisms, even more complicated architectures such as recurrent neural networks that read answers, paragraphs, and questions. Despite their additional complexity, the resulting performance improvements over our straightforward re-ranking mechanisms were only marginal and, oftentimes, we even observed a decline.

2.4 Estimation: Custom Loss/Sub-Sampling

The parameters in $f(\cdot)$ are not trivial to learn. We found that sampling negative (incorrect) and positive (correct) candidates, in combination with a binary classification loss or a regression loss,

was not successful. As a remedy, we propose the following combination of ranking loss and sub-sampling, which proved beneficial in our experiments.

We implement a loss \mathcal{L} , which represents a combination of a pair-wise ranking loss $\mathcal{L}_{\text{rank}}$ and an additional regularization \mathcal{L}_{reg} , in order to train our model. Given two candidate answers i, j with $i \neq j$ for a given question, the binary variables y_i and y_j denote whether the respective candidate answers are correct or incorrect. Then we minimize the following pair-wise ranking loss adapted from Burges et al. (2005), i. e.,

$$\mathcal{L}_{\text{rank}}(x_i, x_j) = \left[y_i - \sigma(f(x_i) - f(x_j)) \right]^2. \quad (2)$$

Here $f(\cdot)$ denotes our previous ranking network and $\sigma(\cdot)$ the sigmoid function. An additional penalty is used to regularize the parameters and prevent the network from overfitting. It is given by

$$\mathcal{L}_{\text{reg}} = \|A\|_1 + \|B\|_1 + \|b_1\|_1 + \|b_2\|_1. \quad (3)$$

Finally, we optimize $\mathcal{L} = \mathcal{L}_{\text{rank}} + \lambda \mathcal{L}_{\text{reg}}$ using

mini-batch gradient descent with λ as a tuning parameter.

We further implement a customized sub-sampling procedure, since the majority of candidate answers generated during training are likely to be incorrect. To address the pair-wise loss during sub-sampling, we proceed as follows: we first generate a list of answer candidates for every question in our training set using the feature extraction and aggregation mechanisms from our re-ranking. Then we iterate through this list and sample a pair of candidate answers (x_i, x_j) if and only if they are at adjacent ranks (i is ranked directly before j i. e., iff $j = i + 1$). We specifically let our training focus on pairs that are originally ranked high, i. e., $j < 4$, and ignore training pairs ranked lower. During inference, we still score all top-10 answer candidates and select the best-scoring answer.

3 Experimental Design

3.1 Content Base and Datasets

Following earlier research, our content base comprises documents from the English Wikipedia. For comparison purposes, we use the same dump as in prior work (e. g., [Chen et al., 2017](#); [Lee et al., 2018](#)).³ We do not use pre-selected documents or other textual content in order to answer questions.

We base our experiments on four well-established datasets.

SQuAD The Stanford Question and Answer Dataset (SQuAD) contains more than 100 000 question-answer-paragraph triples ([Rajpurkar et al., 2016](#)). We use SQuAD_{OPEN}, which ignores the paragraph information.

WikiMovies This dataset contains several thousand question-answer pairs from the movie industry ([Miller et al., 2016](#)). It is designed such that all questions can be answered by a knowledge-base (i. e., Open Movie Database) or full-text content (Wikipedia).

CuratedTREC This dataset is a collection of question-answer pairs from four years of Text Retrieval Conference (TREC) QA challenges ([Baudiš and Šedivý, 2015](#)).

WebQuestions The answers to questions in this dataset are entities in the Freebase

³Downloaded from <https://github.com/facebookresearch/DrQA>

knowledge-base ([Berant et al., 2013](#)). We use the adapted version of [Chen et al. \(2017\)](#), who replaced the Freebase-IDs with textual answers.

3.2 Training Details

Our sourcecode and pre-trained model are available at: <https://github.com/bernhard2202/rankqa>.

RankQA: The information retrieval module is based on the official implementation of [Chen et al. \(2017\)](#).⁴ The same holds true for the pre-trained DrQA-DS model, which we used without alterations. For BERT-QA, we use the uncased BERT base model and fine-tune it for three epochs on the SQuAD training split with the default parameters.⁵

Datasets: We use the training splits of SQuAD, CuratedTREC, WikiMovies, and WebQuestions for training and model selection. In order to balance differently-sized datasets, we use 10 % of the smallest training split for model selection and 90 % for training. For every other dataset, we take the same percentage of samples for model selection and all other samples for training. We monitor the loss on the model selection data and stop training if it did not decrease within the last 10 epochs or after a total of 100 epochs. Finally, we use the model with the lowest error on the model selection data for evaluation. Analogous to prior work, we use the test splits of CuratedTREC, WikiMovies, and WebQuestions, as well as the development split for SQuAD, though only for the final evaluation. In order to account for different characteristics in the datasets, we train a task-specific model individually for every dataset following the same procedure.

Parameters: During training, we use Adam ([Kingma and Ba, 2014](#)) with a learning rate of 0.0005 and a batch size of 256. The hidden layer is set to $m = 512$ units. We set the number of top- n documents to $n = 10$ and the number of top- k candidate answers that are initially generated to $k = 40$. We optimize λ over $\lambda \in \{5 \cdot 10^{-4}, 5 \cdot 10^{-5}\}$. All numerical features are scaled to be within $[0, 1]$. Moreover, we apply an additional log-transformation.

⁴Available at <https://github.com/facebookresearch/DrQA>

⁵Available at <https://github.com/google-research/bert>

	SQuAD _{OPEN}	CuratedTREC	WebQuestions	WikiMovies
Baseline: DrQA (Chen et al., 2017)	29.8	25.4	20.7	36.5
<i>DrQA extensions:</i>				
Paragraph Ranker (Lee et al., 2018)	30.2	35.4	19.9	39.1
Adaptive Retrieval (Kratzwald and Feuerriegel, 2018)	29.6	29.3	19.6	38.4
<i>Other architectures:</i>				
R^3 (Wang et al., 2018)	29.1	28.4	17.1	38.8
DS-QA (Lin et al., 2018)	—	29.1	18.5	—
Min. Context (Min et al., 2018)	34.6	—	—	—
RankQA (general)	34.5	32.4	21.8	43.3
RankQA (task-specific)	35.3	34.7	22.3	43.1
Upper bound: perfect re-ranking for $k = 40$	54.2	65.9	53.8	65.0

Table 2: Exact matches of RankQA compared to DrQA as natural baseline without re-ranking and state-of-the-art systems for neural QA. We use a general model that is trained on all datasets, and a task-specific model that is trained individually for every dataset. The two best results for every dataset are marked in bold.

4 Results

We conduct a series of experiments to evaluate our RankQA system. First, we evaluate the end-to-end performance over the four abovementioned benchmark datasets and compare our system to various other baselines. Second, we show the robustness of answer re-ranking by repeating these experiments with our second implementation, namely BERT-QA. Third, we replicate the experiments of Kratzwald and Feuerriegel (2018) to evaluate the robustness against varying corpus sizes. Fourth, we analyze errors and discuss feature importance in numerical experiments.

During our experiments, we measure the end-to-end performance of the entire QA pipeline in terms of exact matches. That is, we count the fraction of questions for which the provided answer matches one of the ground truth answers exactly. Unless explicitly mentioned otherwise, we refer to the first implementation, namely re-ranking based on the DrQA architecture.

4.1 Performance Improvement from Answer Re-Ranking

Tbl. 2 compares performance across different neural QA systems from the literature. The DrQA system (Chen et al., 2017) is our main baseline as it resembles RankQA without the answer re-ranking step. Furthermore, we compare ourselves against other extensions of the DrQA pipeline such as the *Paragraph Ranker* (Lee et al., 2018) or *Adaptive Retrieval* (Kratzwald and Feuerriegel, 2018). Finally, we compare against other state-of-the-art QA pipelines, namely, R^3 (Wang et al., 2018), *DS-*

QA (Lin et al., 2018), and the *Min. Context* system from Min et al. (2018). For RankQA, we use, on the one hand, a general model that is trained on all four datasets simultaneously. On the other hand, we account for the different characteristics of the datasets and thus employ task-specific models that are trained separately on every dataset.

A direct comparison between DrQA and RankQA demonstrates a performance improvement from up to 7.0 percentage points when using RankQA, with an average gain of 4.9 percentage points over all datasets. Given the identical implementation of information retrieval and machine comprehension, this increase is solely attributable to our answer re-ranking. Our RankQA also outperforms all other state-of-the-art QA systems in 3 out of 4 datasets by a notable margin. This holds true for extensions of DrQA (Paragraph Ranker and Adaptive Retrieval) and other neural QA architectures (R^3 and DS-QA).

This behavior is also observed in the case of the task-specific re-ranking model, which is trained for every dataset individually. Here we achieve performance improvements of up to 9.3 percentage points, with an average performance gain of 5.8 percentage points. The results on the CuratedTREC task deserve further discussion. Evidently, the dataset is particular in the sense that it is very sensitive to specific features. This is confirmed later in our analysis of feature importance and explains why the task-specific RankQA is inferior the general model by a large margin.

Finally, in the last row of Tbl. 2, we provide the results of a perfect re-ranker that always chooses

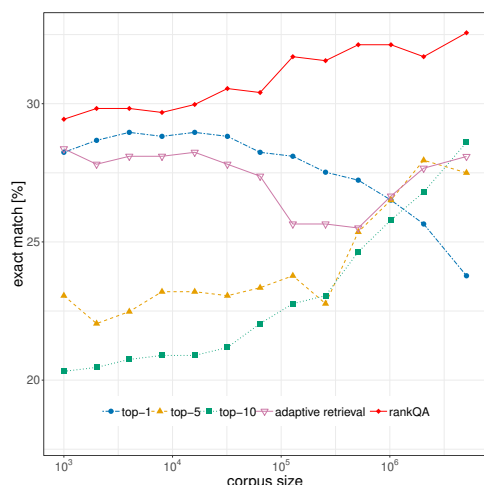


Figure 2: Robustness of answer re-ranking against a variable corpus size. We measure the exact matches for the CuratedTREC dataset while varying the corpus size from one thousand to over five million documents.

the correct answer if present. This system represents an upper bound of the degree to which re-ranking could improve results without changing the information retrieval or machine comprehension models.

4.2 Robustness Check: BERT-QA

In order to demonstrate the robustness of answer re-ranking across different implementations, we repeat experiments from above based on the BERT-QA system. The results are shown in Tbl. 3.

The first row displays the results without answer re-ranking. The second row shows the results after integrating our re-ranking module in the QA pipeline. As one can see, answer re-ranking yields significant performance improvements over all four datasets, ranging between 12.5 and 5.5 percentage points. The last row again lists an upper bound as would have been obtained by a perfect re-ranking system with access to the ground-truth labels. The performance differences between DrQA and BERT can be attributed to the fact that we trained BERT only on the SQuAD dataset, while the pre-trained DrQA model was trained on all four datasets.

4.3 Performance Sensitivity to Corpus Size

Corpora of variable size are known to pose difficulties for neural QA systems. Kratzwald and Feuerriegel (2018) ran a series of experiments in which they monitored the end-to-end performance of different top- n systems (i. e., extracting the answer from the top-10 documents compared

to extracting the answer from the top-1 document only). During the experiments, they increased the size of the corpus from one thousand to over five million documents. They found that selecting $n = 10$ is more beneficial for a large corpus, while $n = 1$ is preferable for small ones. They referred to this phenomenon as a noise-information trade-off: a large n increases the probability that the correct answer is extracted, while a small n reduces the chance that noisy answers will be included in the candidate list. As a remedy, the authors proposed an approach for *adaptive retrieval* that chooses an independent top- n retrieval for every query.

We replicated the experiments of Kratzwald and Feuerriegel (2018)⁶ and evaluated our RankQA system in the same setting, as shown in Fig. 2. We see that answer re-ranking represents an efficient remedy against the noise-information trade-off. The performance of our system (solid red line) exceeds that of any other system configuration for any given corpus size. Furthermore, our approach behaves in a more stable fashion than adaptive retrieval. Adaptive retrieval, like many other recent advancements (e. g., Lee et al., 2018; Lin et al., 2018), limits the amount of information that flows between the information retrieval and machine comprehension modules in order to select better answers. However, RankQA does not limit the information, but directly re-ranks the answers to remove noisy candidates. Our experiments suggest that answer re-ranking is more efficient than limiting the information flow when dealing with variable-size corpora.

4.4 Error Analysis and Feature Importance

We analyze whether our system is capable of keeping the set of correctly answered questions after applying the re-ranking step. Therefore, we measure the fraction of correctly answered questions out of those questions that had been answered correctly before re-ranking. Specifically, we found that the ratio of answers that remained correct varies between 94.6% and 96.1%. Hence, our model does not substantially change initially correct rankings.

Feature importance: Tbl. 4 compares the relative importance of different features. This is measured by training the model with the same pa-

⁶Source code for adaptive retrieval available at: www.github.com/bernhard2202/adaptive-ir-for-qa

	SQuAD _{OPEN}	CuratedTrec	WebQuestions	WikiMovies
Baseline: BERT-QA (no re-ranking)	23.3	19.7	8.2	10.9
RankQA (implementation 2)	35.8	32.0	13.7	20.6
Upper bound: perfect re-ranking for $k = 40$	61.2	66.6	39.6	49.8

Table 3: Exact matches of RankQA based on the BERT-QA pipeline. We show results of the the pipeline without re-ranking, the results obtained by our re-ranking model, and an upper bound (i. e., perfect re-ranking).

	SQuAD _{OPEN}	CuratedTrec	WebQuestions	WikiMovies
Baseline: DrQA (Chen et al., 2017)	29.8	25.4	20.7	36.5
RankQA (general)	34.5	32.4	21.8	43.3
<i>Information Retrieval Features</i>				
RankQA w/o query-document similarity	33.0	29.8	<u>20.6</u>	42.0
RankQA w/o query-paragraph similarity	32.1	32.0	22.0	42.1
RankQA w/o length features	32.9	31.4	22.3	42.6
<i>Machine Comprehension Features</i>				
RankQA w/o linguistic features (POS&NER)	34.4	31.8	21.5	42.3
RankQA w/o ranking features	34.1	31.8	21.4	43.3
RankQA w/o span score	33.4	30.1	21.3	42.3
<i>Feature Aggregation</i>				
RankQA w/o aggregation features	33.6	26.9	<u>18.5</u>	41.5

Table 4: Feature importance (i. e., averaged performance of exact matches on a hold-out sample). We train the general model using the same data, but blind one group of features every time. We underline results that undershoot the baseline and mark results in bold that surpass the general model trained on all features.

rameters and hyperparameters as before; however, we blind one (group of) feature(s) in every round. This was done as follows: when the information retrieval or machine comprehension features were blinded, we also removed the corresponding aggregated features. When omitting aggregation features, we keep the original un-aggregated feature. We show the performance of DrQA (i. e., system without answer re-ranking) and the full re-ranker for the sake of comparison. The original performance increase can only be achieved when all features are included. This has important implications for our approach to properly fusing information from information retrieval and machine comprehension. It suggests that aggregation features are especially informative and that it is not sufficient to use only a subset of those.

We can see that individual datasets reveal a different sensitivity to all feature groups. The CuratedTREC or WebQuestions datasets, for instance, are highly sensitive to some information retrieval features. However, in all cases, the fused combination of features from both information retrieval and machine comprehension is crucial for obtaining a strong performance.

5 Related Work

This work focus on question answering for unstructured textual content in English. Earlier systems of this type comprise various modules such as, for example, query reformulation (e. g., Brill et al., 2002), question classification (Li and Roth, 2006), passage retrieval (e. g., Harabagiu et al., 2000), or answer extraction (Shen and Klakow, 2006). However, the aforementioned modules have been reduced to two consecutive steps with the advent of neural QA.

5.1 Neural Question Answering

Neural QA systems, such as DrQA (Chen et al., 2017) or R^3 (Wang et al., 2018), are usually designed as pipelines of two consecutive stages, namely a module for information retrieval and a module for machine comprehension. The overall performance depends on how many top- n passages are fed into the module for machine comprehension, which then essentially generates multiple candidate answers out of which the one with the highest answer probability score is chosen. However, this gives rise to a noise-information trade-off (Kratzwald and Feuerriegel, 2018). That is, selecting a large n generates many candidate

answers, but increases the probability of selecting the wrong final answer. Similarly, retrieving a small number of top- n passages reduces the chance that the candidate answers contain the correct answer at all.

Resolving the noise-information trade-off in neural QA has been primarily addressed by improving the interplay of modules for information retrieval and machine comprehension. Min et al. (2018) employ sentence-level retrieval in order to remove noisy content. Similarly, Lin et al. (2018) utilize neural networks in order to filter noisy text passages, while Kratzwald and Feuerriegel (2018) forward a query-specific number of text passages. Lee et al. (2018) re-rank the paragraphs *before* forwarding them to machine comprehension. However, none of the listed works introduce answer re-ranking to neural QA.

5.2 Answer Re-Ranking

Answer re-ranking has been widely studied for systems other than neural QA, such as factoid (Severyn and Moschitti, 2012), non-factoid (Moschitti and Quarteroni, 2011), and definitional question answering (Chen et al., 2006). These methods target traditional QA systems that construct answers in non-neural ways, e. g., based on n -gram tiling (Brill et al., 2002) or constituency trees (Shen and Klakow, 2006). However, neural QA extracts an answer directly from text using end-to-end trainable models, rather than constructing it.

With respect to the conceptual idea, closest to our work is the approach of Wang et al. (2017), who use a single recurrent model to re-rank multiple candidate-answers given the paragraphs they have been extracted from. However, this work is different from our RankQA in two ways. First, the authors must read multiple paragraphs in parallel via recurrent neural networks, which limits scalability and the maximum length of paragraphs; see the discussion in Lee et al. (2018). In contrast, our approach is highly scalable and can even be used together with complete corpora and long documents. Second, the authors evaluated their re-ranking in isolation, whereas we integrate our re-ranking into the full QA pipeline where the complete system is subject to extensive experiments.

There are strong theoretical arguments as to why a better fusion of information retrieval and machine comprehension should be benefi-

cial. First, features from information retrieval can potentially be decisive during answer selection (for instance, similarity features or document/paragraph length). Second, answer selection in state-of-the-art systems ignores linguistic features that are computed during the machine comprehension phase (e. g., DrQA uses part-of-speech and named entity information). Third, although some works aggregate scores for similar answers (e. g., Lee et al., 2018; Wang et al., 2017), the complete body information is largely ignored during aggregation. This particularly pertains to, e. g., how often and with which original rank the top- n answers were generated.

6 Conclusion

Our experiments confirm the effectiveness of a three-stage architecture in neural QA. Here answer re-ranking is responsible for bolstering the overall performance considerably: our RankQA represents the state-of-the-art system for 3 out of 4 datasets. When comparing it to corresponding two-staged architecture, answer re-ranking can be credited with an average performance improvement of 4.9 percentage points. This performance was even rendered possible with a light-weight architecture that allows for the efficient fusion of information retrieval and machine comprehension features during training. Altogether, RankQA provides a new, strong baseline for future research on neural QA.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPUs used for this research.

References

- Petr Baudiš and Jan Šedivý. 2015. *Modeling of the Question Answering Task in the YodaQA System*. In *International Conference on Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 222–228.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. *Semantic Parsing on Freebase from Question-Answer Pairs*. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1533–1544.
- Eric Brill, Susan Dumais, and Michele Banko. 2002. *An analysis of the AskMSR question-answering sys-*

- tem. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 257–264.
- Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. [Learning to rank using gradient descent](#). In *International Conference on Machine Learning (ICML)*, pages 89–96.
- Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. [Reading Wikipedia to Answer Open-Domain Questions](#). In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1870–1879.
- Yi Chen, Ming Zhou, and Shilong Wang. 2006. [Reranking answers for definitional QA using language modeling](#). In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1081–1088. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.
- Sanda M. Harabagiu, Dan I. Moldovan, Marius Păca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Corina R. Gîrju, Vasile Rus, and Paul Morrescu. 2000. [FALCON: Boosting Knowledge for Answer Engines](#). In *Text Retrieval Conference (TREC)*, pages 479–488.
- Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*.
- Bernhard Kratzwald and Stefan Feuerriegel. 2018. [Adaptive Document Retrieval for Deep Question Answering](#). In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 576–587.
- Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. 2018. [Ranking Paragraphs for Improving Answer Recall in Open-Domain Question Answering](#). In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 565–569.
- Xin Li and Dan Roth. 2006. [Learning question classifiers: The role of semantic information](#). *Natural Language Engineering*, 12(03):229–249.
- Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. [Denoising Distantly Supervised Open-Domain Question Answering](#). In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1736–1745.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. [Building a Large Annotated Corpus of English: The Penn Treebank](#). *Computational Linguistics*, 19(2):313–330.
- Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. [Key-Value Memory Networks for Directly Reading Documents](#). In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1400–1409.
- Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. [Efficient and Robust Question Answering from Minimal Context over Documents](#). In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1725–1735.
- Alessandro Moschitti and Silvia Quarteroni. 2011. [Linguistic kernels for answer re-ranking in question answering systems](#). *Information Processing and Management*, 47(6):825–842.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ Questions for Machine Comprehension of Text](#). In *Empirical Methods in Natural Language Processing (EMNLP) Language Processing*, pages 2383–2392.
- Aliaksei Severyn and Alessandro Moschitti. 2012. [Structural relationships for large-scale learning of answer re-ranking](#). In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 741–750.
- Dan Shen and Dietrich Klakow. 2006. [Exploring correlation of dependency relation paths for answer extraction](#). In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 889–896.
- Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. [R³: Reinforced Reader-Ranker for Open-Domain Question Answering](#). In *Association for the Advancement of Artificial Intelligence (AAAI)*.
- 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](#). In *International Conference on Learning Representations (ICLR)*.
- Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019a. [End-to-End Open-Domain Question Answering with BERTserini](#). In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL, Demo)*.
- Wei Yang, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019b. [Data Augmentation for BERT Fine-Tuning in Open-Domain Question Answering](#). *arXiv preprint arxiv: 1904.06652*.