Incremental Improvement of a Question Answering System by Re-ranking Answer Candidates using Machine Learning

Michael Barz and Daniel Sonntag

Abstract We implement a method for re-ranking top-10 results of a state-of-the-art question answering (QA) system. The goal of our re-ranking approach is to improve the answer selection given the user question and the top-10 candidates. We focus on improving deployed QA systems that do not allow re-training or when re-training comes at a high cost. Our re-ranking approach learns a similarity function using n-gram based features using the query, the answer and the initial system confidence as input. Our contributions are: (1) we generate a QA training corpus starting from 877 answers from the customer care domain of T-Mobile Austria, (2) we implement a state-of-the-art QA pipeline using neural sentence embeddings that encode queries in the same space than the answer index, and (3) we evaluate the QA pipeline and our re-ranking approach using a separately provided test set. The test set can be considered to be available after deployment of the system, e.g., based on feedback of users. Our results show that the system performance, in terms of top-n accuracy and the mean reciprocal rank, benefits from re-ranking using gradient boosted regression trees. On average, the mean reciprocal rank improves by 9.15%.

1 Introduction

In this work, we examine the problem of incrementally improving deployed QA systems in an industrial setting. We consider the domain of customer care of a wireless network provider and focus on answering frequent questions (focussing on the

Michael Barz

German Research Center for Artificial Intelligence, Saarland Informatics Campus, D3 2, 66123 Saarbrücken, Germany e-mail: `michael.barz@dfki.de`

Saarbrücken Graduade School of Computer Science

Daniel Sonntag

German Research Center for Artificial Intelligence, Saarland Informatics Campus, D3 2, 66123 Saarbrücken, Germany e-mail: `daniel.sonntag@dfki.de`

long tail of the question distribution [2]). In this setting, the most frequent topics
are covered by a separate industry-standard chatbot based on hand-crafted rules by
dialogue engineers. Our proposed process is based on the augmented cross-industry
standard process for data mining [23] (augmented CRISP data mining cycle). In
particular, we are interested in methods for improving a model after its deployment
through re-ranking of the initial ranking results. In advance, we follow the steps of
the CRISP cycle towards deployment for generating a state-of-the-art baseline QA
model. First, we examine existing data (*data understanding*) and prepare a corpus
for training (*data preparation*). Second, we implement and train a QA pipeline using
state-of-the-art open source components (*modelling*). We perform an evaluation us-
ing different amounts of data and different pipeline configurations (*evaluation*), also
to understand the nature of the data and the application (*business understanding*).
Third, we investigate the effectiveness and efficiency of re-ranking in improving our
QA pipeline after the *deployment* phase of CRISP. Adaptivity after deployment is
modelled as *(automatic) operationalisation* step with external reflection based on,
e.g., user feedback. This could be replaced by introspective meta-models that al-
low the system to enhance itself by metacognition [23]. The QA system and the
re-ranking approach are evaluated using a separate test set that maps actual user
queries from a chat-log to answers of the QA corpus. Sample queries from the eval-
uation set with one correct and one incorrect sample are shown in Table 1.

With this work, we want to answer the question whether a deployed QA system
that is difficult to adapt and that provides a top-10 ranking of answer candidates,
can be improved by an additional re-ranking step that corresponds to the opera-
tionalisation step of the augmented CRISP cycle. It is also important to know the
potential gain and the limitations of such a method that works on top of an exist-
ing system. We hypothesise that our proposed re-ranking approach can effectively
improve ranking-based QA systems.

2 Related Work

The broad field of QA includes research ranging from retrieval-based [29, 8, 15, 9]
to generative [21, 20], as well as, from closed-domain [10, 16] to open-domain QA
[20, 13, 18, 7]. We focus on the notion of improving an already deployed system.

For QA dialogues based on structured knowledge representations, this can be
achieved by maintaining and adapting the knowledgebase [26, 24, 25]. In addition,
[23] proposes metacognition models for building self-reflective and adaptive AI sys-
tems, e.g., dialogue systems, that improve by introspection. Buck et al. present a
method for reformulating user questions: their method automatically adapts user
queries with the goal to improve the answer selection of an existing QA model [5].

Other works suggest humans-in-the-loop for improving QA systems. Savenkov
and Agichtein use crowdsourcing for re-ranking retrieved answer candidates in a
real-time QA framework [19]. In Guardian, crowdworkers prepare a dialogue sys-
tem based on a certain web API and, after deployment, manage actual conversa-

Table 1 Sample queries with a correct and an incorrect answer option according to our test set. We report the answers' rank of the baseline model that we used for our re-ranking experiments.

User Query	Correct Answer	Incorrect Answer
Bekomme ich bei Vertragsverlängerung ein neues Handy? (*Do I get a new phone when extending my contract?*)	Ab wann Sie Ihre Rufnummern verlängern können und welche Angebote bei einer Vertragsverlängerung auf Sie warten, sehen Sie in Ihrem persönlichen Kundenportal Mein T-Mobile [...] (*In your online customer area of T-Mobile, you can see when you can continue your telephone numbers and which offers await you after extending your contract [...]*) **(rank 1)**	Suchen Sie ein neues Gerät, das genau Ihre Bedürfnisse und Anforderungen erfüllt? Sie wollen rechtzeitig über Neuerungen informiert werden? [...] (*You are looking for a new device that satisfies all your requirements? You want to get recent news? [...]*) **(rank 5)**
tarife ohne bi**m**dung (*plans without bi**m**ding contract*) – `misspelled`	Wenn Sie bereits ein Handy besitzen und nur eine Simkarte benötigen, haben wir genau das Richtige für Sie: die Klax-SIM. [...] (*If you own a new phone and all you need is a SIM card, we got exactly the right offer: the Klax-SIM. [...]*) **(rank 3)**	Eine Übersicht über unsere aktuellen My Mobile Handytarife inklusive aller wichtigen Details finden Sie auf der Tarifseite. [...] (*An overview of our current service plans with all important details can be found on our website. [...]*) **(rank 1)**
Kosten für vertragübernahme (*costs for a contract transfer*)	Sie können Verträge an andere Personen übergeben, zusammenlegen oder trennen. Die Kosten belaufen sich auf [...] Ausführliche Informationen zum Thema finden Sie in den FAQ. (*You can transfer, join and split contracts from and to other persons. The costs are [...] More detailed information can be found in our FAQ.*) **(rank 10)**	Ein Zukauf von Freiminuten ist nicht möglich und bei unseren aktuellen Tarifen auch nicht notwendig, da Freiminuten hier unlimitiert sind. (*You cannot buy additional minutes. However, that's not required with our plans, because minutes are unlimited.*) **(rank 1)**
Kreditkarte (*credit card*)	Eine Zahlung mittels Kreditkarte ist selbstverständlich bei uns möglich. Sollten Sie Ihre Zahlungsart auf Kreditkarte ändern oder Ihre Daten aktualisieren wollen, können Sie dies direkt über unseren LiveChat veranlassen. [...] (*Of course, you can pay with your credit card. If you want to change your payment settings to credit card or if you want to update your data, you can do so using our LiveChat. [...]*) **(rank 7)**	Die Änderung Ihrer Kreditkartendaten ist zu Ihrer Sicherheit nur telefonisch bei der Serviceline unter [...] (*For security reasons, you can change your credit card data via phone using our service hotline at [...] only*) **(rank 1)**
Hallo, ich möchte ein iPhone 7 kaufen (Ratenzahlung). Ich hab schon ein Vertrag (bis 09.2017)..wenn ich das verlängern möchte muss ich die Raten von meine altes Handy weiter zahlen? Lg (*Hello, I'd like to buy an iPhone 7 (paying by instalments). I have a contract (till 09/2017)..if I want to extend it, do I need to pay the remaining rates for my old phone? Kind regards*)	Ratenzahlungen oder Stundungen bei offenen Rechnungsbeträgen bietet T-Mobile NICHT an [...] (*T-Mobile does NOT offer payment by instalments or deferred payments for outstanding bill amounts.*) **(not in top-10)**	Mit der Umstellung auf LTE hat sich nichts am Geschwindigkeitsprofil inklusive der erreichbaren Maximalgeschwindigkeiten Ihres aktuellen Tarifes geändert. [...] (*The transition to LTE (4G) does not affect the maximum data transfer rate of your present service plan.*) **(rank 1)**

tions with users [12]. EVORUS learns to select answers from multiple chatbots via crowdsourcing [11]. The result is a chatbot ensemble excels the performance of each individual chatbot. Williams et al. present a dialogue architecture that continuously learns from user interaction and feedback [27].

We propose a re-ranking algorithm similar to [19]: we train a similarity model using n-gram based features of QA pairs for improving the answer selection of a retrieval-based QA system.

3 Question Answering System

We implement our question answering system using state-of-the-art open source components. Our pipeline is based on the Rasa natural language understanding (NLU) framework [4] which offers two standard pipelines for text classification: *spacy_sklearn* and *tensorflow_embedding*. The main difference is that *spacy_sklearn* uses Spacy[1] for feature extraction with pre-trained word embedding models and Scikit-learn [17] for text classification. In contrast, the *tensorflow_embedding* pipeline trains custom word embeddings for text similarity estimation using TensorFlow [1] as machine learning backend. Figure 1 shows the general structure of both pipelines. We train QA models using both pipelines with the pre-defined set of hyper-parameters. For *tensorflow_embedding*, we additionally monitor changes in system performance using different epoch configurations[2]. Further, we compare the performances of pipelines with or without a spellchecker and investigate whether model training benefits from additional user examples by training models with the three different versions of our training corpus including no additional samples (kw), samples from 1 user (kw+1u) or samples from 2 users (kw+2u) (see section Corpora). All training conditions are summarized in Table 2. Next, we describe the implementation details of our QA system as shown in Figure 1: the spellchecker module, the subsequent pre-processing and feature encoding, and the text classification. We include descriptions for both pipelines.

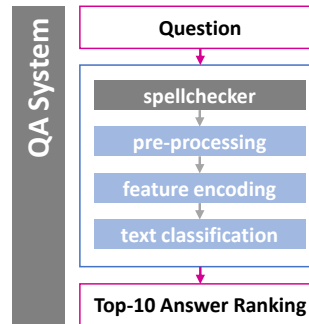**Table 2** Considered configurations for QA pipeline training.

	spacy_sklearn	*tensorflow_embedding*
parameters	default	default with epochs $\in$ $\{10, 50, 100, 300, 600\}$
spellchecking		yes, no
training corpus		kw, kw+1u, kw+2u

Spellchecker We address the problem of frequent spelling mistakes in user queries by implementing an automated spell-checking and correction module. It is based

---

[1] https://spacy.io/

[2] https://rasa.com/docs/nlu/components/#intent-classifier-tensorflow-embedding

Fig. 1 The basic configuration of the QA pipeline, which is a part of our complete QA system architecture with the re-ranking algorithm.

on a Python port[3] of the SymSpell algorithm[4] initialized with word frequencies for German[5]. We apply the spellchecker as first component in our pipeline.

Pre-Processing and Feature Encoding. The *spacy_sklearn* pipeline uses Spacy for pre-processing and feature encoding. Pre-processing includes the generation of a Spacy `document` and tokenization using their German language model `de_core_news_sm` (v2.0.0). The feature encoding is obtained via the `vector` function of the Spacy document that returns the mean word embedding of all tokens in a query. For German, Spacy provides only a simple dense encoding of queries (no proper word embedding model).

The pre-processing step of the *tensorflow_embedding* pipeline uses a simple whitespace tokenizer for token extraction. The tokens are used for the feature encoding step that is based on Scikit-learn's `CountVectorizer`. It returns a bag of words histogram with words being the tokens (1-grams).

Text Classification. The *spacy_sklearn* pipeline relies on Scikit-learn for text classification using a support vector classifier (SVC). The model confidences are used for ranking all answer candidates; the top-10 results are returned.

Text classification for *tensorflow_embedding* is done using TensorFlow with an implementation of the StarSpace algorithm [28]. This component learns (and later applies) one embedding model for user queries and one for the answer id. It minimizes the distance between embeddings of QA training samples. The distances between a query and all answer ids are used for ranking.

---

[3] `https://github.com/mammothb/symspellpy`

[4] `https://github.com/wolfgarbe/SymSpell`

[5] German 50k: `https://github.com/hermitdave/FrequencyWords`

3.1 Corpora

In this work, we include two corpora: one for training the baseline system and another for evaluating the performance of the QA pipeline and our re-ranking approach. In the following, we describe the creation of the training corpus and the structure of the test corpus. Both corpora have been anonymised.

Training Corpus. The customer care department provides 877 answers to common user questions. Each answer is tagged with a variable amount of keywords or keyphrases ($M = 3.81$, $SD = 5.92$), 3338 in total. We asked students to augment the training corpus with, in total, two additional natural example queries. This process can be scaled by crowdsourcing for an application in productive systems that might include more answers or that requires more sample question per answer or both. The full dataset contains, on average, 5.81 sample queries per answer totalling in 5092 queries overall. For model training, all questions (including keywords) are used as input with the corresponding answer as output. We generated three versions of the training corpus: keywords only (`kw`, $n = 3338$), keywords with samples from 1 user (`kw+1u`, $n = 4215$) and keywords with samples from 2 users (`kw+2u`, $n = 5092$).

Evaluation Corpus. The performance of the implemented QA system and of our re-ranking approach is assessed using a separate test corpus. It includes 3084 real user requests from a chat-log of T-Mobile Austria, which are assigned to suitable answers from the training corpus (at most three). The assignment was performed manually by domain experts of the wireless network provider. We use this corpus for estimating the baseline performance of the QA pipeline using different pipeline configurations and different versions of the training corpus. In addition, we use the corpus for evaluating our re-ranking approach per cross-validation: we regard the expert annotations as offline human feedback. The queries in this corpus contain a lot of spelling mistakes. We address this in our QA pipeline generation by implementing a custom spell-checking component.

4 Baseline Performance Evaluation

We evaluate the baseline model using all training configurations in Table 2 to find a well-performing baseline for our re-ranking experiment. We use the evaluation corpus as reference data and report the top-1 to top-10 accuracies and the mean reciprocal rank for the top-10 results (MRR@10) as performance metrics. For computing the top-n accuracy, we count all queries for which the QA pipeline contains a correct answer on rank 1 to n and divide the result by the number of test queries. The MRR is computed as the mean of reciprocal ranks over all test queries. The reciprocal rank for one query is defined as $RR = \frac{1}{rank}$: The RR is 1 if the correct answer is ranked first, 0.5 if it is at the second rank and so on. We set RR to zero, if the answer is not contained in the top-10 results.

Fig. 2 Performance metrics in terms of top-1 to top-10 accuracy and MRR@10 of both QA pipelines for different pipeline configurations and training corpora.

pipeline	spell check	corpus	Accuracy (top-n)										MRR @10
			1	2	3	4	5	6	7	8	9	10	
spacy-sklearn	no	kw	0.073	0.129	0.169	0.199	0.225	0.250	0.270	0.288	0.305	0.317	0.139
		kw+1u	0.095	0.156	0.194	0.223	0.246	0.265	0.287	0.304	0.317	0.332	0.161
		kw+2u	0.099	0.158	0.202	0.226	0.253	0.274	0.290	0.304	0.320	0.330	0.165
	yes	kw	0.095	0.150	0.205	0.238	0.265	0.286	0.304	0.322	0.342	0.356	0.167
		kw+1u	0.117	0.186	0.241	0.280	0.313	0.333	0.355	0.368	0.387	0.404	0.198
		kw+2u	0.127	0.203	0.250	0.295	0.327	0.354	0.375	0.393	0.412	0.428	0.212
tensorflow embedding (300 epochs)	no	kw	0.125	0.192	0.240	0.279	0.308	0.332	0.345	0.351	0.360	0.364	0.198
		kw+1u	0.156	0.227	0.277	0.318	0.347	0.365	0.381	0.394	0.404	0.413	0.233
		kw+2u	0.195	0.273	0.328	0.360	0.383	0.404	0.418	0.434	0.443	0.454	0.274
	yes	kw	0.194	0.276	0.326	0.363	0.390	0.412	0.424	0.435	0.445	0.456	0.275
		kw+1u	0.190	0.277	0.330	0.366	0.401	0.419	0.432	0.442	0.454	0.464	0.276
		kw+2u	0.180	0.274	0.332	0.375	0.408	0.435	0.451	0.463	0.478	0.492	0.275

Fig. 3 Performance metrics in terms of top-1 to top-10 accuracy and MRR@10 for the *tensorflow_embedding* pipeline with spell-checking for different training corpora and a different number of training `epochs`.

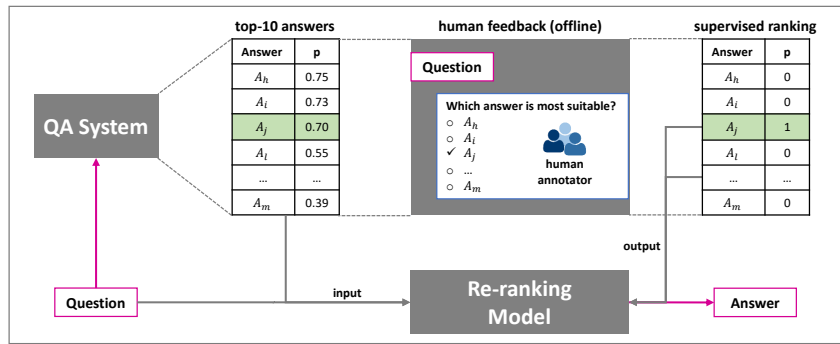| corpus	epochs	Accuracy (top-n)										MRR @10
		1	2	3	4	5	6	7	8	9	10	
kw	10	0.193	0.261	0.298	0.321	0.334	0.349	0.361	0.371	0.378	0.387	0.255
	50	0.196	0.263	0.302	0.324	0.342	0.354	0.366	0.377	0.387	0.396	0.258
	100	0.181	0.265	0.312	0.345	0.365	0.380	0.396	0.405	0.416	0.425	0.259
	300	0.194	0.276	0.326	0.363	0.390	0.412	0.424	0.435	0.445	0.456	0.275
	600	0.145	0.220	0.284	0.326	0.362	0.387	0.400	0.415	0.430	0.439	0.232
kw+1u	10	0.189	0.253	0.294	0.320	0.334	0.350	0.364	0.376	0.386	0.396	0.252
	50	0.204	0.292	0.336	0.364	0.384	0.400	0.413	0.421	0.431	0.441	0.281
	100	0.235	0.318	0.369	0.401	0.428	0.445	0.459	0.474	0.485	0.493	0.316
	300	0.190	0.277	0.330	0.366	0.400	0.419	0.432	0.442	0.454	0.464	0.276
	600	0.183	0.276	0.333	0.373	0.400	0.425	0.442	0.459	0.472	0.485	0.275
kw+2u	10	0.189	0.268	0.311	0.337	0.359	0.377	0.389	0.400	0.413	0.425	0.263
	50	0.213	0.287	0.331	0.363	0.387	0.405	0.424	0.440	0.451	0.463	0.288
	100	0.208	0.299	0.353	0.391	0.417	0.438	0.454	0.466	0.478	0.494	0.296
	300	0.180	0.274	0.332	0.375	0.408	0.435	0.451	0.463	0.478	0.492	0.275
	600	0.196	0.300	0.356	0.393	0.421	0.438	0.455	0.470	0.486	0.498	0.292

Results. Figure 2 shows the accuracy and MRR values for all conditions. We only restrict *tensorflow_embedding* to the default number of `epochs` which is 300. At the corpus level, we can observe that the accuracy and the MRR increase when training with additional user annotations for all pipeline configurations. For example, the *spacy_sklearn* pipeline without spell-checking achieves a top-10 accuracy of 0.317 and a MRR of 0.139 when using the `kw` training corpus with keywords only. Both measures increase to 0.33 and 0.165, respectively, when adding two natural queries for training. In some cases, adding only 1 user query results in slightly better scores. However, the overall trend is that more user annotations yield better results.

In addition, we observe performance improvements for pipelines that use our spell-checking component when compared to the default pipelines that do not make use of it: The *spacy_sklearn* `kw+2u` condition performs 9.8% better, the *tensorflow_embedding* `kw+2u` condition performs 3.8% better, in terms of top-10 accuracy. We can observe similar improvements for the majority of included metrics. Similar to the differentiation by corpus, we can find cases where spell-checking reduces the performance for a particular measure, against the overall trend.

Overall, the *tensorflow_embedding* pipelines perform considerably better than the *spacy_sklearn* pipeline irrespective of the remaining parameter configuration: the best performing methods are achieved by the *tensorflow_embedding* pipeline with spell-checking. Figure 3 sheds more light on this particular setting. It provides performance measures for all corpora and for different number of `epochs` used for model training. Pipelines that use 300 `epochs` for training range among the best

for all corpora. When adding more natural user annotations, using 100 `epochs` achieves similar or better scores, in particular concerning the top-10 accuracy and the MRR. Re-ranking the top-10 results can only improve the performance in QA, if the correct answer is among the top-10 results. Therefore, we use the *tensor-flow_embedding* pipeline with spellchecking, 100 `epochs` and the full training corpus as baseline for evaluating the re-ranking approach.

5 Re-Ranking Approach

Fig. 4 Complete QA system architecture including the re-ranking model. The re-ranking model is trained using manually annotated data for generating a supervised/ideal ranking result for the top-10 answers from the QA system. Features are extracted from the user question and a particular answer candidate. At inference time, the re-ranking model is used to improve the initial top-10 ranking.

Our re-ranking approach compares a user query with the top-10 results of the baseline QA system. In contrast to the initial ranking, our re-ranking takes the content of the answer candidates into account instead of encoding the user query only. Our algorithm compares the text of the recent user query to each result. We include the answer text and the confidence value of the baseline system for computing a similarity estimate. Finally, we re-rank the results by their similarity to the query (see Algorithm 1).

We consider a data-driven similarity function that compares linguistic features of the user query and answer candidates and also takes into account the confidence of the baseline QA system. This similarity estimate shall enhance the baseline by using an extended data and feature space, but without neglecting the learned patterns of the baseline system. The possible improvement in top-1 accuracy is limited by the top-10 accuracy of the baseline system (49.4%), because our re-ranking cannot choose from the remaining answers. Figure 4 shows how the re-ranking model is

---

Algorithm 1: Re-Ranking Algorithm

Input: a user query q; the corresponding list of top-10 results R including an answer a and the baseline confidence c;

Output: an updated ranking R'

begin

   $R' \longleftarrow []$;

   foreach $(c, a) \in R$ **do**

      $c' \longleftarrow similarity(q, a, c)$;

      $R'.append((c', a))$;

   `// sort R' by confidences c', descending`

   $sort(R')$;

   return R'

---

connected to the deployed QA system: it requires access to its in- and outputs for the additional ranking step.

We consider the gradient boosted regression tree for learning a similarity function for re-ranking similar to [19]. The features for model training are extracted from pre-processed query-answer pairs. Pre-processing includes tokenization and stemming of query and answer and the extraction of uni-, bi- and tri-grams from both token sequences[6]. We include three distance metrics as feature: the Jaccard distance, the cosine similarity[7], and the plain number of n-gram matches between n-grams of a query and an answer.

6 Re-Ranking Performance Evaluation

We compare our data-driven QA system with a version that re-ranks resulting top-10 candidates using the additional ranking model. We want to answer the question whether our re-ranking approach can improve the performance of the baseline QA pipeline after deployment. For that, we use the evaluation corpus ($n = 3084$) for training and evaluating our re-ranking method using 10-fold cross-validation, i.e., 90% of the data is used for training and 10% for testing with 10 different train-test splits.

The training and testing procedure per data split of the cross-validation is shown in Algorithm 2. For each sample query q in the train set C_{train}, we include the correct answer a^+ and one randomly selected negative answer candidate a^- for a balanced model training. We skip a sample, if the correct answer is not contained in the top-10 results: we include 49.4% of the data (see top-10 accuracy of the baseline QA model in Figure 3). The baseline QA model r and the trained re-ranking method r' are applied to all sample queries in the test set C_{test}. Considered performance metrics

---

[6] We use default word tokenizer, Snowball stemmer and n-gram extraction of the nltk toolkit [3]

[7] We use the implementation for Jaccard distance and cosine similarity as found in the following Github gist: gaulinmp/similarity_example.ipynb

---

Algorithm 2: Evaluation Procedure (per Data Split)

Input: a train- and test split of the evaluation corpus C_{train}, C_{test}, each including QA-pairs as tuples (q, a^+); the pre-trained baseline QA model for initial ranking r and the untrained re-ranking model r'.

Output: evaluation metrics.

begin
   // training of the re-ranking model
   $X \longleftarrow []$;
   $y \longleftarrow []$;
   foreach $(q, a^+) \in C_{train}$ **do**
      $R \longleftarrow r.rank(q)$;   // R contains top-10 results
      if $a^+ \notin R$) **then**
         continue with next QA pair
      else
         // add positive sample
         $c^+ \longleftarrow R[a^+]$;   // confidence for a^+
         $X.append(features\_from(q, a^+, c^+))$;
         $y.append(1)$;
         // add negative sample
         $a^- \longleftarrow$ random $a \in R \setminus a^+$;
         $c^- \longleftarrow R[a^-]$;
         $X.append(features\_from(q, a^-, c^-))$;
         $y.append(0)$;
   $r'.train(X, y)$;
   // evaluation of the re-ranking model
   $results \longleftarrow \emptyset$;
   foreach $(q, a^+) \in C_{test}$ **do**
      $R \longleftarrow r.rank(q)$;   // top-10 baseline ranking
      $R' \longleftarrow r'.rank(q, R)$;   // **apply re-ranking**
      $results.append(R')$;
   return $compute\_metrics(results)$

---

are computed using the re-ranked top-10 *results*. We repeat the cross-validation 5 times to reduce effects introduced by the random selection of negative samples. We report the average metrics from 10 cross-validation folds and the 5 repetitions of the evaluation procedure.

Results. The averaged cross-validation results of our evaluation, in terms of top-n accuracies and the MRR@10, are shown in Table 3: the top-1 to top-9 accuracies improve consistently. The relative improvement decreases from 14.83% for the top-1 accuracy to 1.68% for the top-9 accuracy. The top-10 accuracy stays constant, because the re-ranking cannot choose from outside the top-10 candidates. The MRR improves from 0.296 to 0.323 (9.15%).

Table 3 Performance metrics of the baseline QA pipeline and our re-ranking method ($n = 3084$).

Metric	Method		Relative Improvement
	Baseline QA	*Re-Ranking*	
top-1 accuracy	0.208	0.239	14.83%
top-2 accuracy	0.299	0.334	11.84%
top-3 accuracy	0.353	0.384	8.99%
top-4 accuracy	0.391	0.415	6.31%
top-5 accuracy	0.417	0.44	5.59%
top-6 accuracy	0.438	0.459	4.83%
top-7 accuracy	0.454	0.471	3.74%
top-8 accuracy	0.466	0.48	3.02%
top-9 accuracy	0.478	0.486	1.68%
top-10 accuracy	0.494	0.494	0.00%
MRR@10	0.296	0.323	9.15%

7 Discussion

Our results indicate that the accuracy of the described QA system benefits from our re-ranking approach. Hence, it can be applied to improve the performance of already deployed QA systems that provide a top-10 ranking with confidences as output. However, the performance gain is small, which might have several reasons. For example, we did not integrate spell-checking in our re-ranking method which proved to be effective in our baseline evaluation. Further, the re-ranking model is based on very simple features. It would be interesting to investigate the impact of more advanced features, or models, on the ranking performance (e.g., word embeddings [14] and deep neural networks for learning similarity functions [8, 15]). Nevertheless, as can be seen in examples 1, 2 and 4 in Table 1, high-ranked but incorrect answers are often meaningful with respect to the query: the setting in our evaluation is overcritical, because we count incorrect, but meaningful answers as negative result. A major limitation is that the re-ranking algorithm cannot choose answer candidates beyond the top-10 results. It would be interesting to classify whether an answer is present in the top-10 or not. If not, the algorithm could search outside the top-10 results. Such a meta-model can also be used to estimate weaknesses of the QA model: it can determine topics that regularly fail, for instance, to guide data labelling for a targeted improvement of the model, also known as active learning [22], and in combination with techniques from semi-supervised learning [9, 6].

Data labelling and incremental model improvement can be scaled by crowdsourcing. Examples include the parallel supervision of re-ranking results and targeted model improvement as human oracles in an active learning setting. Results from crowd-supervised re-ranking allows us to train improved re-ranking models [19, 11], but also a meta-model that detects queries which are prone to error. The logs of a deployed chatbot, that contain actual user queries, can be efficiently analysed using such a meta-model to guide the sample selection for costly human data augmentation and creation. An example of a crowdsourcing approach that could be applied to our QA system and data, with search logs can be found in [2].

8 Conclusion

We implemented a simple re-ranking method and showed that it can effectively improve the performance of QA systems after deployment. Our approach includes the top-10 answer candidates and confidences of the initial ranking for selecting better answers. Promising directions for future work include the investigation of more advanced ranking approaches for increasing the performance gain and continuous improvements through crowdsourcing and active learning.

References

1. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
2. Michael S. Bernstein, Jaime Teevan, Susan Dumais, Daniel Liebling, and Eric Horvitz. Direct answers for search queries in the long tail. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, page 237, New York, New York, USA, 2012. ACM Press.
3. Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.
4. Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. Rasa: Open Source Language Understanding and Dialogue Management. 12 2017.
5. Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. Ask the Right Questions: Active Question Reformulation with Reinforcement Learning. *CoRR*, abs/1705.0, 2017.
6. Joseph Chee Chang, Aniket Kittur, and Nathan Hahn. Alloy: Clustering with Crowds and Computation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, pages 3180–3191, New York, New York, USA, 2016. ACM Press.
7. Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879. ACL, 2017.
8. Arpita Das, Harish Yenala, Manoj Chinnakotla, and Manish Shrivastava. Together we stand: Siamese Networks for Similar Question Retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 378–387, Stroudsburg, PA, USA, 2016. Association for Computational Linguistics.
9. Bhuwan Dhingra, Danish Danish, and Dheeraj Rajagopal. Simple and Effective Semi-Supervised Question Answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 582–587. ACL, 2018.
10. Mihail Eric and Christopher D Manning. Key-Value Retrieval Networks for Task-Oriented Dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49, Saarbrücken, Germany, 8 2017. Association for Computational Linguistics.
11. Ting-Hao 'Kenneth' Huang, Joseph Chee Chang, and Jeffrey P. Bigham. Evorus: A Crowd-powered Conversational Assistant Built to Automate Itself Over Time. 1 2018.

12. Ting-Hao Kenneth Huang, Walter S. Lasecki, and Jeffrey P. Bigham. Guardian: A Crowd-Powered Spoken Dialog System for Web APIs. In *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015.
13. Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 1: Long Papers*, pages 1601–1611. ACL, 2017.
14. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. 1 2013.
15. Shervin Minaee and Zhu Liu. Automatic Question-Answering Using A Deep Similarity Neural Network. 8 2017.
16. Shereen Oraby, Pritam Gundecha, Jalal Mahmud, Mansurul Bhuiyan, and Rama Akkiraju. "How May I Help You?": Modeling Twitter Customer ServiceConversations Using Fine-Grained Dialogue Acts. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces - IUI '17*, pages 343–355, New York, New York, USA, 2017. ACM Press.
17. F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
18. Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. ACL, 2016.
19. Denis Savenkov and Eugene Agichtein. CRQA: Crowd-Powered Real-Time Automatic Question Answering System. *Fourth AAAI Conference on Human Computation and Crowdsourcing*, 2016.
20. Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. 7 2015.
21. Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. A Survey of Available Corpora for Building Data-Driven Dialogue Systems. 12 2015.
22. Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.
23. Daniel Sonntag. On Introspection, Metacognitive Control and Augmented Data Mining Live Cycles. jul 2008.
24. Daniel Sonntag. Introspection and adaptable model integration for dialogue-based question answering. In *IJCAI*, pages 1549–1554, 2009.
25. Daniel Sonntag. *Ontologies and Adaptivity in Dialogue for Question Answering*, volume 4 of *Studies on the Semantic Web*. AKA and IOS Press, Heidelberg, first edition, 2010.
26. Daniel Sonntag, Ralf Engel, Gerd Herzog, Alexander Pfalzgraf, Norbert Pfleger, Massimo Romanelli, and Norbert Reithinger. Smartweb handheld - multimodal interaction with ontological knowledge bases and semantic web services. In *Artifical Intelligence for Human Computing, ICMI 2006 and IJCAI 2007 International Workshops, Banff, Canada, November 3, 2006, Hyderabad, India, January 6, 2007, Revised Seleced and Invited Papers*, pages 272–295, 2007.
27. Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. Hybrid Code Networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 665–677, Stroudsburg, PA, USA, 2017. Association for Computational Linguistics.
28. Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. Starspace: Embed all the things! *CoRR*, abs/1709.03856, 2017.
29. Xiaobing Xue, Jiwoon Jeon, and W. Bruce Croft. Retrieval models for question and answer archives. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '08*, page 475, New York, New York, USA, 2008. ACM Press.