Learning from On-Line User Feedback in Neural Question Answering on the Web

Bernhard Kratzwald
ETH Zurich
Zurich, Switzerland
bkratzwald@ethz.ch

Stefan Feuerriegel
ETH Zurich
Zurich, Switzerland
sfeuerriegel@ethz.ch

ABSTRACT

Question answering promises a means of efficiently searching web-based content repositories such as Wikipedia. However, the systems of this type most prevalent today merely conduct their learning once in an offline training phase while, afterwards, all parameters remain static. Thus, the possibility of improvement over time is precluded. As a consequence of this shortcoming, question answering is not currently taking advantage of the wealth of feedback mechanisms that have become prominent on the web (e. g., buttons for liking, voting, or sharing).

This is the first work that introduces a question-answering system for (web-based) content repositories with an on-line mechanism for user feedback. Our efforts have resulted in QApedia – a framework for on-line improvement based on shallow user feedback. In detail, we develop a simple feedback mechanism that allows users to express whether a question was answered satisfactorily or whether a different answer is needed. Even for this simple mechanism, the implementation represents a daunting undertaking due to the complex, multi-staged operations that underlie state-of-the-art systems for neural questions answering. Another challenge with regard to web-based use is that feedback is limited (and possibly even noisy), as the true labels remain unknown. We thus address these challenges through a novel combination of neural question answering and a dynamic process based on distant supervision, asynchronous updates, and an automatic validation of feedback credibility in order to mine high-quality training samples from the web for the purpose of achieving continuous improvement over time.

Our QApedia framework is the first question-answering system that continuously refines its capabilities by improving its now dynamic content repository and thus the underlying answer extraction. QApedia not only achieves state-of-the-art results over several benchmarking datasets, but we further show that it successfully manages to learn from shallow user feedback, even when the feedback is noisy or adversarial. Altogether, our extensive experimental evaluation, with more than 2,500 hours of computational experiments, demonstrates that a feedback mechanism as simple as a binary vote (which is widespread on the web) can considerably improve performance when combined with an efficient framework for continuous learning.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313661

CCS CONCEPTS

• **Information systems** → **Question answering**; • **Computing methodologies** → *Lifelong machine learning*; *Online learning settings.*

KEYWORDS

Question answering; User feedback; Distant Supervision; On-line learning

ACM Reference Format:
Bernhard Kratzwald and Stefan Feuerriegel. 2019. Learning from On-Line User Feedback in Neural Question Answering on the Web. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313661

1 INTRODUCTION

Question-answering (QA) systems promise an efficient way to search content repositories, as both questions and answers are presented in natural language [39]. Use cases can include the vast majority of freely-available materials that have been developed on the web. Hence, question answering has been developed for traditional knowledge bases and knowledge graphs [e. g., 1, 2, 14, 42, 46] but also for unstructured open-domain content such as Wikipedia [3, 8, 11] and social media [7], as well as various domain-specific applications including Stack Overflow [4].

Content-based question answering widely assumes that answer extraction mechanisms are static; see the systems in [8, 11] as illustrative examples. This is in contrast to QA for knowledge bases, where a key objective involves building up the knowledge base itself [12, 17], as well as continuously updating the underlying decision rules for answering [2]. However, in the context of QA for linguistic content, such dynamic mechanisms are, to the best of our knowledge, non-existent.

This paper suggests a framework for content-based QA in order to bolster QA systems with dynamic capabilities: our framework aids continuous learning on the basis of a simple mechanism for user feedback. In fact, user feedback plays a prominent role in web-based systems, as a large proportion of websites now offer tools for social interaction. Examples include buttons for liking or sharing content (e. g., as on Instagram), as well as voting with an array of categorical emotions (e. g., as on Facebook) [26]. There have been previous attempts to utilize user interactions in classical information retrieval [5, 16, 28] or knowledge bases [1], and yet there is a dearth of content-based QA systems that leverage user interactions.

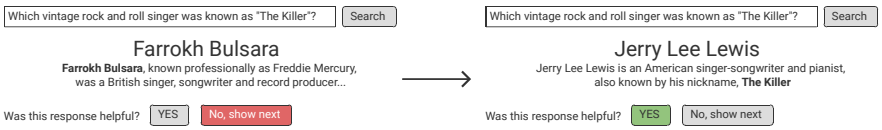

Figure 1: Illustrative sketch of shallow user feedback in question-answering for natural language. A simple credibility check is often sufficient in order to judge whether an answer makes sense in the given context.

Extending content-based QA with user feedback promises several benefits. When interacting with a QA system, the correct answer is unknown to a user, and yet it is fairly easy to judge whether an answer makes sense in the given context (see Fig. 1). In addition, shallow feedback in the form of a binary vote can be collected at low cost. It is especially low-cost in comparison to reverting to a human annotator in order to retrieve the correct label. Finally, the prevalence of feedback mechanisms on the web ensures that such user interactions have become widely intuitive.

Contributions: This work proposes QApedia: a neural question-answering framework for encyclopedic content that continuously improves on the basis of on-line user feedback. To the best of our knowledge, QApedia represents the first content-based QA system that improves over time. Our feedback framework advances the status quo of static QA systems while being specifically tailored to web-based settings:

(1) **Feedback for dynamic knowledge.** Content on the web is subject to considerable time variability and, hence, a QA system must adapt to this dynamic nature. To facilitate this, we develop an effective feedback mechanism so that the abilities of the QA system can successfully continue to improve over time. Our framework directly incorporates user feedback in an end-to-end loop: collected feedback is fed back into the system in an on-line fashion. As a key challenge during implementation, we must overcome the problem of catastrophic forgetting that is known in neural networks and thus also neural QA. For this reason, we develop a tailored form of distant supervision with asynchronous updates.
(2) **Shallow feedback.** We only require shallow user feedback in the form of a simple up or down vote, which is nowadays common on the web. We specifically refrain from asking users to report the exact answers, as users might not know these answers or else be reluctant to report them; instead, it is sufficient for our framework to receive a simple credibility check. Receiving only limited feedback – and not necessarily the correct solution – requires a specialized adaptation of distant supervision to our setting.
(3) **Noisy and adversarial feedback.** User feedback in web-based settings is often noisy or even adversarial. Our framework must therefore be designed so that, despite errors in user feedback, it maintains its performance (or even continues to improve) and is thus especially robust. This is achieved by incorporating a validation procedure, based on knowledge mining, during which the credibility of user feedback is checked.

Our findings demonstrate that our QApedia framework successfully manages to learn from on-line feedback. It not only adapts to the feedback provided in the on-line setting, but it also maintains the abilities it has acquired through previous training, thus overcoming the issue of catastrophic forgetting. Our results yield a considerable improvement: the user feedback ensures that the performance over time no longer remains static but, even when evaluating the QA system with question-answer pairs from a different domain, the number of correct answers continues to increase over time on the order of 10–20 percentage points. For instance, in one dataset, fewer than 60,000 user interactions with shallow feedback were sufficient to double the percentage of exact answers. These performance improvements are even maintained in the case of noisy and adversarial feedback. Furthermore, catastrophic forgetting in a naïve QA system decreases the ratio of exact answers by 5 percentage points, whereas our QApedia framework largely maintains the original performance.

2 RELATED WORK

Question Answering: Question answering can be divided into two main paradigms, namely systems that operate in relation to structured knowledge and those that rely upon (primarily unstructured) textual content (or both, as in [15]).

QA systems for structured knowledge [e. g., 1, 2, 14, 42, 46] derive answers from knowledge bases, ontologies, or knowledge graphs. Structured knowledge bases augment web search and sometimes even serve as substitutes; see, for instance, Wikidata or Google Knowledge Graph. Explicit structures entail the benefit of simplifying the process of answer extraction, yet they are incomplete and limited to rigid (and often pre-defined) schemata and, therefore, lack the same flexibility as running text.

QA systems for content in natural language [e. g., 9, 11, 17, 32, 43] overcome some of the drawbacks of raw knowledge bases, as they extract answers directly from an underlying corpus of unstructured text documents. Hence, they find widespread application in mining web-based content such as Wikipedia or other online encyclopedias [3, 8, 11]. The content-based approach greatly contributes to overall flexibility, especially when such systems leverage the growing body of knowledge in web-based content repositories. Hence, QA systems for (web-based) content repositories constitute the focus of this work. Yet prior systems for question answering for content repositories have been designed as static systems: all decision rules are determined once and are static thereafter, thus curbing any form of continuous improvement.

Neural Question Answering for Content: Content-based QA systems commonly proceed through multiple phases [25]: they first select a subset of documents (or paragraphs) that are considered relevant and then extract the final answer. Answer extraction has traditionally been based on linguistic rules or pattern matching [13, 21, 32, 38], whereas deep neural networks [11, 18, 36, 43] have evolved only recently as the state-of-the-art. This is later reflected by our implementation, in which we combine several of the recent

advances from neural question answering, such as attention [41, 44] and adaptive retrieval [24], in order to obtain a reference system that represents the status quo.

Learning from User Feedback: Improving (i. e., learning) from user feedback is a well-studied problem in, for instance, information retrieval and web-based search [5, 16, 28]. Applications in these disciplines include user feedback that is leveraged in order to improve conversational agents [37], assess the relevance of answers in search queries for knowledge graphs [40], or identify incorrect queries to databases [19]. While [35] makes use of shallow binary feedback for machine translation, it does not account for noisy or adversarial feedback.

Closest to our research is the seminal work by Abujabal et al. [1] that distinguishes the relevance of a set of potential answers based on user feedback in the context of question answering for a structured knowledge base. Yet the latter work addresses a different objective and, in contrast to our research, thus entails marked differences. First, the aforementioned feedback mechanism from [1] is tailored to knowledge bases, while we contribute to the specific methodological foundation of content-based question answering. Second, the authors' work integrates user feedback into ontology-based representations, while we extract answers from natural language based on deep neural networks. Third, the methodology in [1] utilizes user feedback at strategic intervention points (i. e., when decision rules are unknown), whereas we develop a framework in which on-line feedback is directly integrated end-to-end. That is, user feedback is collected continuously across potentially all user interactions and then directly incorporated at the most effective point of the QA system such that the overall performance of the QA task improves. Furthermore, our work presents – to the best of our knowledge – the first attempt in the field of question answering that adapts to the particular characteristics of user feedback on the web, namely the possibility of errors and even malicious feedback; therefore, we evaluate our proposed framework with user feedback that is noisy or even adversarial.

Never-Ending Learning: In never-ending learning [10, 31], models continue to learn new tasks and/or improve upon existing tasks over time, even after their initial estimation. Never-ending learning represents an especially challenging undertaking in combination with neural networks, as neural networks are inherently prone to so-called catastrophic forgetting [23]. Catastrophic forgetting describes the phenomenon whereby models forget what they have learned earlier; i. e., while learning new tasks, the performance on tasks that have already been trained decreases dramatically.

In prior literature, applications of never-ending learning to question answering are scarce. To the best of our knowledge, there is only one such approach, the NEQA system [1]. This system is designed to address the specific needs of knowledge bases and is thus not applicable to content-based QA as in our research. More precisely, the NEQA system accumulates new rules on the basis of which it extract answers from a knowledge base. In contrast, we pursue a different objective as we develop a data-driven approach for updating the deep neural network that is nowadays common in content-based QA: both training data and model parameters are treated as dynamic components. For this purpose, we need to derive a specific framework involving distant supervision combined with asynchronous updating.

Distant Supervision: Distant supervision (i. e., enriching weakly labeled data with labels that are potentially noisy) has been previously used for neural question answering, but for purposes other than aiding never-ending learning. Previous use cases include, for example, locating correct answers in text passages [29] or for data augmentation [11, 20]. In this work, we contribute two-fold: (1) in contrast to prior works, we apply distant supervision to on-line feedback mechanisms for question answering in relation to content repositories. (2) We suggest a nascent setting in which on-line user feedback can be subject to errors, thus necessitating a novel feedback loop whereby user feedback is validated against the current content repository as a prerequisite for performing distant supervision.

3 QApedia: CONTINUOUS IMPROVEMENT FROM USER FEEDBACK

The architecture of our proposed QApedia framework (see Fig. 2) consists of three modules: a *content base* stores the knowledge for answering questions. Examples include, e. g., Wikipedia or other content repositories in natural language. This module also manages the datasets that are used for learning the question-answering capabilities. The *question-answering module* provides the actual system that takes a question as input and then suggests an answer. Its implementation adheres to the state-of-the-art of neural QA as detailed in Sec. 4. The module for *continuous improvement* represents our core contribution and ensures that dynamic learning takes place. It thus receives feedback through user interactions and, on this basis, updates the content base and, subsequently, the neural network inside the question-answering module.

3.1 Feedback Mechanism through User Interactions

3.1.1 Motivation Behind Shallow User Feedback. The QApedia framework achieves its dynamic learning based on user interactions that are simple in nature: a user is merely asked to assess the overall fit of an answer in the given context and should then respond with a simple up- or down-vote. This binary feedback system was specifically chosen as it promises inherent benefits. First, shallow user feedback can be collected at low cost. Accordingly, it does not require expert users or rigorous fact-checking through which the correct answer is determined. Instead, we strive for a simple integration whereby the context of an answer is assessed, and we thus expect that user feedback might include a small portion of erroneous assessments. Similar feedback mechanisms have become prevalent in web-based use cases [34, 35], e. g., when users are asked to classify whether a translation was helpful. Hence, users should be familiar with this arrangement, which ensures straightforward use.

Our design, based on shallow feedback, also comes with limitations: if the QA system comes up with the wrong answer, it will likely receive a negative vote. However, it does not necessarily obtain the correct answer, which can be used as a label. This poses challenges – beyond those that accompany web-based use – that we must carefully address in our design.

Another challenge arises from the web-based nature of the user inputs: there is no guarantee that user feedback is correct. Some users might be better at judging the suitability of an answer in a

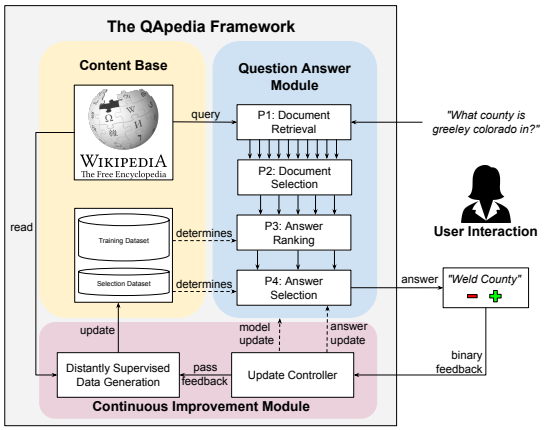

Figure 2: Architecture of our QApedia framework.

given context than others. Some users might, however, just click a random feedback option or even test the system's capability by intentionally responding with an erroneous answer. Hence, the system must adapt to noisy (i. e., partially random) and adversarial (i. e., explicitly wrong) feedback.

3.1.2 Feedback Collection. The QA system takes a question q as input and then returns a triplet $\langle q, a, p \rangle$, where a refers to the answer and p to the surrounding paragraph that can be used for assessing the fit (see the example presentation in Fig. 1). Let ψ denote the (optional) feedback, i. e., an up-vote or a down-vote. Depending on the feedback, the system proceeds as follows:

- *Positive feedback* is directly passed to the module for continuous improvement. The module first performs a credibility check to validate the answer a with an up-vote against the content base. It then generates a new data sample $\langle q, a, p \rangle$ (or, more precisely, $\langle q, a, p^* \rangle$ with an improved p^* through distant supervision as described in Section 3.3) and, based on this, updates the datasets in the content base. The updated training sets are then used at regular intervals for re-estimating the neural network inside the QA module.
- *Negative feedback* triggers a process in which the current answer is rephrased and an alternative answer a' is displayed to the user. For this purpose, the QA module is queried in order to generate the next-best but different answer. We achieve this by choosing the next answer in the ranked list and ensuring that it is not a duplicate of the original answer. Eliminating duplicates is necessary, since the candidate answers during answer ranking are often identical. Users can repeatedly provide negative feedback, where ρ denotes the maximum frequency that we use in our experimental evaluation, whereas such a limit would not be enforced in practice.

The user feedback is directly fed into the content base through distant supervision and thus follows a human-in-the-loop process.

3.2 Content Base

Our QApedia framework performs question answering for content repositories. As part of our experiments, we later adhere to earlier research [3, 8, 11] and utilize Wikipedia as a widespread baseline. Without loss of generality, our framework can also be used in relation to all other knowledge bases that contain unstructured materials in (English) natural language.

Our model estimation further relies upon two datasets with labeled questions, answers, and paragraphs. These datasets are time-dependent and thus dynamic in order to enable continuous learning. Then, the training dataset $\mathcal{D}^t_{\text{train}}$ at time t is needed for estimating the parameters inside the question-answering module. A second dataset $\mathcal{D}^t_{\text{selection}}$ at time t is used for model selection. Both datasets contain triplets $\langle q, a, p \rangle$ with a question q, an answer a, and the corresponding paragraph p (i. e., in which the answer a is located).

It is important to point out that the dynamic, time-dependent nature of our datasets $\mathcal{D}^t_{\text{train}}$ and $\mathcal{D}^t_{\text{selection}}$ is integral for incorporating user feedback at runtime. This allows us to include user feedback in the form of new data samples. Finally, this time dependency is in stark contrast to prior literature on content-based QA [e. g., 11, 43], which builds upon corpora that are static. In such examples, the QA models are trained once during an off-line phase, whereas we implement the datasets for training and model selection as dynamic components updated through on-line feedback.

3.3 Continuous Improvement Module

The continuous improvement module is responsible for three tasks. First, user feedback is not directly fed into the content base but subject to a customized process with *distant supervision*. This entails an internal credibility check of the user feedback against the content base in order to deal with feedback that is noisy or adversarial. Second, the credibility check produces aggregated evidence with high-quality data samples that is eventually integrated into the content base. Third, the actual updates of the QA module can be triggered at arbitrary time intervals, for which we draw upon *asynchronous model updates*. As part of this, all trainable parameters inside the QA module are re-estimated based on the current state of the content base.

Prior work on distant supervision in question answering [11, 20] assumes that the provided answer to a question is always correct, whereas a web-based setting does not allow for this assumption and we thus adapt to incorrect feedback via credibility validation.

3.3.1 Credibility Validation. QApedia improves under noisy and adversarial feedback thanks to a careful design: we do not directly integrate triples $\langle q, a, p \rangle$ with a positive up-vote into the training data but instead rely upon a novel validation procedure with distant supervision, which filters erroneous or malicious feedback. As a credibility check, we mine so-called evidence $\mathcal{E} = \{p_1, \ldots, p_n\}$ from the content base that supports the user feedback. Only if this evidence surpasses a certain threshold do we treat the user feedback as credible.

Evidence is mined as follows. We utilize the QA system – more precisely, the mechanisms for document retrieval and document selection – in order to retrieve text passages $\mathcal{E}$ from the content base that are likely to contain the answer to question q. The resulting evidence $\mathcal{E}$ is further processed by a set of filtering rules, so that we obtain only a subset $\mathcal{E}_{\text{filtered}} \subseteq \mathcal{E}$. Without difficulty, we can reject all passages that fail to contain the assumed answer a; i. e., $a \notin p$. These passages do not provide any strong evidence for a

Algorithm 1: Updating procedure for continuous improvement of QA systems from shallow user feedback.

Input: User feedback ψ for question-answer-paragraph triple $\langle q, a, p \rangle$, where $\psi \in \{0, \uparrow, \downarrow\}$ denotes either an up-vote, down-vote, or no vote

$\mathcal{E}_{\text{filtered}} \leftarrow \emptyset$
if $\psi = \downarrow$ **then**
  Trigger answer selection to refine the displayed answer
else if $\psi = \uparrow$ **then**
  Initialize evidence $\mathcal{E} \leftarrow$ document_selection(q) by querying for q
  for $p \in \mathcal{E}$ **do**
    if $a \in p$ **and** $len(p) > min\_len$ **and** $NE(q) \subseteq NE(p)$ **then**
      Score the evidence $s \leftarrow score(p, q, a)$ based on a sliding window
      if $s \geq 2$ **then**
        $\mathcal{E}_{\text{filtered}} \leftarrow \mathcal{E}_{\text{filtered}} \cup \{\langle q, s \rangle\}$;
      end
    end
  end
  if $|\mathcal{E}_{\text{filtered}}| >$ threshold τ **then**
    Determine best paragraph $p^* \leftarrow \max \mathcal{E}_{\text{filtered}}$ in the evidence (with the score from document retrieval as a tie breaker)
    Determine randomly which corpus $c \leftarrow corp(q, a)$ to update
    if $c =$ train **then**
      $\mathcal{D}^{t+1}_{\text{train}} = \mathcal{D}^{t}_{\text{train}} \cup \{\langle q, a, p^* \rangle\}$;
    else
      $\mathcal{D}^{t+1}_{\text{selection}} = \mathcal{D}^{t}_{\text{selection}} \cup \{\langle q, a, p^* \rangle\}$;
    end
  end
end

being an answer of q. Furthermore, we filter passages that are very short, since we noticed that these commonly contain summarizing statements or headlines but are rarely extensive enough to contain the answer. Accordingly, we check that the paragraph length $len(p)$ exceeds a minimum length (here we use min_len = 25). Following [11, 20], we further filter paragraphs that do not contain all of the named entities that were found in the question.[1] Let $NE(p)$ refer to the named entities – we then ensure that $NE(q) \subseteq NE(p)$. Finally, we score the quality of each of the remaining passages by counting the bi-gram overlap between the question and a 15-word window around the found answer in the passage. We reject all passages where the score is below a threshold. In our experiments, we follow [11] and enforce at least two overlaps of bi-grams. All remaining passages $\mathcal{E}_{\text{filtered}}$ are considered as strong evidence for the correctness of the feedback.

The decision of whether or not an item of feedback is credible is made based on the number of remaining text passages, i. e., $|\mathcal{E}_{\text{filtered}}| \geq \tau$. The hyperparameter τ controls the trade-off between a fast improvement (i. e., more samples will be generated) on the one hand, and a more restrictive filtering of noise on the other hand (i. e., more feedback will be rejected). We demonstrate the effects of rigorous filtering in our experiments on noisy and adversarial users.

3.3.2 Data Generation via Distant Supervision. If the user feedback is found to be credible, we do not add $\langle q, a, p \rangle$ directly into our content base; instead, we generate a new data sample $\langle q, a, p^* \rangle$ with replaced paragraph p^* from the evidence. The aim behind this is to find a high-quality paragraph p^* which contains more information than the original paragraph p that received the up-vote. To achieve this, we make direct use of the aforementioned credibility validation and the evidence set $\mathcal{E}_{\text{filtered}}$ accumulated in that phase. In detail, we use the highest scoring text passage $p^* \in \max_p \mathcal{E}_{\text{filtered}}$, i. e., we take the paragraph with the highest bi-gram overlap. If there are multiple passages with the same overlap, we use the retrieval score as a tiebreaker. This results in a new data sample $\langle q, a, p^* \rangle$, which is added to the dataset either for training or model selection, i. e.,

$$\mathcal{D}^{t+1}_{\text{train}} \leftarrow \mathcal{D}^{t}_{\text{train}} \cup \{\langle q, a, p^* \rangle\} \quad \text{or} \tag{1}$$

$$\mathcal{D}^{t+1}_{\text{selection}} \leftarrow \mathcal{D}^{t}_{\text{selection}} \cup \{\langle q, a, p^* \rangle\}. \tag{2}$$

The choice is made randomly, while maintaining the original ratio of the splits between training and model selection set (i. e., 90 % vs. 10 %). Furthermore, we assert that no question is in both datasets at the same time and that every dataset contains no duplicate question-answer pairs.

3.3.3 Asynchronous Model Updates. At desired time steps t, the continuous improvement module invokes an asynchronous model update. Here the idea is to use the current training sets from the content base, i. e., $\mathcal{D}^{t}_{\text{train}}$ and $\mathcal{D}^{t}_{\text{selection}}$ at time t in order to re-estimate the neural QA system. We use the above training data $\mathcal{D}^{t}_{\text{train}}$ with mini-batch gradient descent for 45 epochs. After each epoch, we evaluate the performance on the selection dataset $\mathcal{D}^{t}_{\text{selection}}$. Finally, we select the model with the highest-scoring performance on the selection dataset and use it to replace our current model.

The combination of distant supervision and data-driven learning of model parameters proves to be very robust against noisy or even adversarial feedback, as it provides two lines of defense. The first line of defense is our method for generating new data samples, since it filters most of the noise before it can enter the system. The second line of defense is the update mechanism itself. Its data-driven nature tolerates a small fraction of noise by definition, since noisy gradients are averaged out as long as the corpus contains enough correct samples. This is in strong contrast to a rule-based system, where even one wrong rule is guaranteed to generate wrong answers.

4 THE QUESTION-ANSWERING MODULE

The question-answering module within our framework for learning from on-line feedback provides the functional core that is responsible for extracting the displayed answers from the web-based content repository. Its structure is designed such that it combines recent advances in the field of neural question answering that are regarded as state-of-the-art in order to yield a reference implementation. Specifically, it builds upon attention mechanisms [41, 44], adaptive retrieval [24], and neural answer extraction at paragraph level [11, 43], concepts that have been proposed only recently. Consistent with the literature, the question-answering module operates along four phases (see Fig. 2), which we summarize briefly in the following: In **(P1) Document Retrieval**, all documents from the content repository are scored based on their relevance to the question. In **(P2) Document Selection**, after ranking all documents by their relevance, only a small subset of documents is selected, while all others are discarded. In **(P3) Answer Ranking**, a deep neural network is used to suggest the locations of candidate answers. In **(P4) Answer Selection**, finally, the probabilities are used

[1] We used the Stanford CoreNLP toolkit: https://nlp.stanford.edu/software/

in order to determine the best answer over all paragraphs and documents. The resulting answer is also the one that is presented to the user or leveraged for additional computations inside the proposed framework for continuous improvement from on-line feedback.

4.1 P1: Document Retrieval

During information retrieval, all documents in the content repository are ranked based on their relevance to the input question q. Our actual implementation is analogous to the neural question answering from [11], which is widely used as a baseline. In detail, the approach from [11] represents each question q and all documents as vectors with hashed bigram counts [45] that are weighted by tf-idf. Then a dot product between the vectors of the question and each document is computed in order to obtain the relevance scores. This process is further accelerated by constructing all vectors using an inverted index in order to allow for fast retrieval.

4.2 P2: Document Selection

During document selection, the content repository is filtered for N documents that entail the highest relevance scores. It is often suboptimal to select only a single document (i. e., $N = 1$) since the recall can improve by choosing a larger subset (i. e., $N > 1$). However a larger subset also increases the proportion of noise and thus the possibility of eventually selecting a wrong answer. As a remedy, we follow the adaptive approach in [24], where the actual size n is determined based on a trainable classifier. Such an adaptive document selection is known to be especially suited to settings in which the size of the content repository is variable [24]. This adaptive selection represents an important element in our continuous learning framework, in which the dataset for training is intentionally designed to be dynamic and thus grows over time.

4.3 P3: Answer Ranking

The selected documents are further split into paragraphs and, in order to obtain greater accuracy in practice, all subsequent steps are performed at paragraph-level, as suggested by [11]. The objective is then to determine the probability of a candidate answer based on the following deep neural network. For this purpose, we compute the probability distributions over all start and end positions. These are given by p^s and p^e, respectively.

The architecture consists of an embedding layer, separate encoding layers for both the question and paragraphs, as well as a subsequent prediction layer. The network architecture adapts a simplified version of the document reader that was used in [11]; however, it is extended by additional encoding layers similar to [43] in order to yield a good predictive power while maintaining a reasonable runtime.

Input: The neural network for answer extraction receives input in the form of a paragraph $p = \langle p_1, \ldots, p_l \rangle$ of length l and a question $q = \langle q_1, \ldots, q_k \rangle$ of length k, where $p_1, \ldots, p_l$ and $q_1, \ldots, q_k$ denote the tokens therein.

Embedding Layer: The embedding layer maps each token onto a vector representation.[2] A question q is then given by an $n \times k$ matrix $\boldsymbol{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k]$, where n is the embedding dimension and $\boldsymbol{q}_i$ (in bold) denotes the embedding vector that corresponds to the original token q_i, for $i = 1, \ldots, k$. Analogously, we translate the paragraph p into an $n \times l$ matrix $\boldsymbol{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_l]$. Following [11, 27], we further compute a paragraph-aligned representation of the question $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l]$ using question-to-query attention $\boldsymbol{u}_i = \sum_{j=1}^{k} \alpha_{i,j}\, \boldsymbol{q}_j$ with

$$\alpha_{i,j} = \frac{\exp\left(f(\boldsymbol{p}_i)^T\, f(\boldsymbol{q}_j)\right)}{\sum_{j'=1}^{k} \exp\left(f(\boldsymbol{p}_i)^T\, f(\boldsymbol{q}_{j'})\right)} \quad \text{for } i = 1, \ldots, l, \tag{3}$$

where $f(\cdot)$ refers to a fully-connected neural network layer that we implemented with ReLU activation.

Encoding Layer for Paragraphs: A multi-layered bidirectional long short-term memory (BiLSTM) is used in order to encode the previous embeddings. That is, we apply BiLSTMs to both the paragraph and the paragraph-aligned representations in order to concatenate their hidden states along the sequence of tokens, i. e.,

$$\boldsymbol{H}^P = \text{BiLSTM}_1(\boldsymbol{P}) \quad \text{and} \quad \boldsymbol{H}^U = \text{BiLSTM}_2(\boldsymbol{U}), \tag{4}$$

where $\boldsymbol{H}^P$ and $\boldsymbol{H}^U$ are matrices of size $2mh$-by-l with h being the hidden size of the LSTM and m the number of bi-directional layers (hence the 2). Notably, these are separate models – namely, BiLSTM_1 and BiLSTM_2 – which do not share any parameters. Both are finally combined into a joint encoding $\boldsymbol{G}$ of the paragraph by stacking the hidden states, their element-wise interactions [43], and exact match features [11]. More specifically, this is accomplished with the help of an additional multi-layer BiLSTM given by

$$\boldsymbol{G} = \text{BiLSTM}_3\left(\left[\boldsymbol{H}^P; \boldsymbol{H}^U; \boldsymbol{H}^P \odot \boldsymbol{H}^U; \boldsymbol{z}\right]\right), \tag{5}$$

where $\boldsymbol{G}$ is again a $2mh \times l$ matrix, $\odot$ refers to the element-wise multiplication and exact match features are represented by an l-dimensional binary vector $\boldsymbol{z} = [z_1, \ldots, z_l]$ with $z_i = \mathbb{1}_{p_i \in q}$. In other words, z_i equals one if the i-th token of a paragraph is included in the question.

Encoding Layer for Questions: Questions are encoded by feeding their embeddings into a multilayer BiLSTM network, yielding a matrix of size $2mh$-by-k that is given by

$$\boldsymbol{H}^Q = \text{BiLSTM}_4(\boldsymbol{Q}). \tag{6}$$

In order to obtain a representation $\boldsymbol{r}$ of questions that is independent of their length, we use so-called self-attention [44] which sums over all k tokens via

$$\boldsymbol{r} = \sum_{i=1}^{k} \beta_i\, \boldsymbol{h}_i^Q \quad \text{with} \quad \beta_i = \frac{\exp(\boldsymbol{w}^T\, \boldsymbol{q}_i)}{\sum_{i'=1}^{k} \exp(\boldsymbol{w}^T\, \boldsymbol{q}_{i'})}, \tag{7}$$

where $\boldsymbol{w}$ is a trainable parameter and $\boldsymbol{h}_i^Q$ is the i-th column vector of $\boldsymbol{H}^Q$.

Prediction Layer: Following [11], we obtain scores proportional ($\propto$) to the probabilities for which each token represents the start and the end position of a candidate answer. This is computed via

$$\boldsymbol{p}^s \propto \exp(\boldsymbol{G}^T \boldsymbol{W}_s \boldsymbol{r}) \quad \text{and} \quad \boldsymbol{p}^e \propto \exp(\boldsymbol{G}^T \boldsymbol{W}_e \boldsymbol{r}), \tag{8}$$

respectively, such that $\boldsymbol{W}_s$ and $\boldsymbol{W}_e$ denote weight matrices of size $2mh$-by-$2mh$ that can be trained. For any subspan a of a paragraph p that ranges from token a_s to a_e, we can now determine the probability of answering a question: this probability is proportional to the dot product between $p^s_{a_s}$ and $p^e_{a_e}$. Notably, we refrain from

[2] We utilized pre-trained, case-sensitive Glove embeddings, the so-called common crawl from http://nlp.stanford.edu/data/glove.840B.300d.zip.

normalizing these scores in order to ensure comparability across paragraphs and even documents.

4.4 P4: Answer Selection

The phase for answer selection aims at determining the best answer out of all candidates, i. e., $\max_a \mathbb{P}(a \mid q, p)$. Hence, one simply optimizes over the joint probabilities of start and end positions that are given by $\boldsymbol{p}^s$ and $\boldsymbol{p}^e$, respectively. Consistent with [11], an additional constraint is enforced in order to ensure that the end token comes after the start token, as well as guaranteeing that the final answer has a maximum length (here: 15 tokens, as suggested in [11]). This yields the following optimization problem

$$\max_{a_s, a_e} \left(p^s_{a_s}\right)^T p^e_{a_e} \quad \text{s. t.} \quad a_s \leq a_e \leq a_s + 15, \tag{9}$$

which selects the final answer that is displayed to the user.

In the case of negative feedback, the answer selection module is triggered to refine the answer by selecting the next best scoring answer that is different from the down-voted answer shown before.

5 EXPERIMENTAL SETUP

5.1 Content Repository

Our QA system draws upon the English-language Wikipedia[3] as the central source of knowledge in order to answer questions. The English-language Wikipedia currently consists of more than five million documents containing more than eight million words and, therefore, provides an excellent basis for question answering that spans various disciplines and topics. Wikipedia is freely available and has an active community that keeps it up-to-date. By following a content-based paradigm for QA, we directly query Wikipedia articles and extract answers from these. We also use Wikipedia when validating the credibility of user feedback, a process in which we aggregate evidence in a distantly-supervised fashion that should validate the correctness of the feedback.

5.2 Question-Answer Corpora

We use two different question-answer corpora during our experiments in order to highlight the dynamic nature (i. e., dependent on time step t): one to initialize our system (at $t = 0$) and one to simulate user feedback after deployment (for $t > 0$); see Fig. 3. This split into two different phases represents an experimental setup that is especially challenging since both corpora stem from separate domains, each with a question-answer style that is distinctly different. Hence, it allows us to benchmark both the ability of continuous learning in relation to new question-answer pairs and its robustness against forgetting when samples stem from the initial domain.

During *initialization time* ($t = 0$), we use the Stanford Question Answering Corpora (SQuAD) [33] to initialize the training data $\mathcal{D}^{t=0}_{\text{train}}$ and the selection data $\mathcal{D}^{t=0}_{\text{selection}}$ within the framework, and to estimate the model parameters. SQuAD includes more than 100,000 question-answer-paragraph samples with paragraphs exclusively extracted from Wikipedia. We keep a hold-out fraction of about 10,000 SQuAD samples, which we call $\mathcal{D}_{\text{forgetting}}$, in order to re-evaluate the performance over time and test whether our system is prone to catastrophic forgetting. Our hypothesis is that a system that is initialized on SQuAD and robust against catastrophic forgetting does not decrease in its initial performance on the SQuAD corpus (e. g., $\mathcal{D}_{\text{forgetting}}$).

After *deployment* ($t > 1$), we simulate user interactions on another corpus with a substantially different question-answer style. To show the robustness of our results, we run three independent experiments using the TriviaQA [20] corpus, the WikiMovies [30] corpus, and a third corpus that combines questions from TriviaQA (25 %) and WikiMovies (50 %) with the WebQuestions [6] corpus (25 %). In all three experiments, these corpora are split into a simulation part $\mathcal{D}^{t>0}_{\text{sim}}$ which provides the samples that represent our simulated users and a hold-out part $\mathcal{D}_{\text{learn}}$ in order to evaluate the extent to which continuous learning is achieved. Our hypothesis is: a system with our continuous improvement module increases its initial performance on $\mathcal{D}_{\text{learn}}$ due to learning and, at the same time, its performance remains robust against forgetting when evaluated on $\mathcal{D}_{\text{forgetting}}$.

5.3 Simulation of User Interactions

User interactions are simulated based on the following routine:

Initialization: use the SQuAD corpus to initialize the training data $\mathcal{D}^{t=0}_{\text{train}}$ and selection data $\mathcal{D}^{t=0}_{\text{selection}}$

Deployment: repeat for time steps $t = 1, \ldots$

- **(a) Simulation:** simulate $N_\phi = 10,000$ user interactions with up- and down-votes based on the corpus $\mathcal{D}^t_{\text{sim}}$ that is used after deployment
- **(b) Distant supervision:** validate the credibility of the user feedback and, if necessary, generate new data samples from the user feedback
- **(c) Dataset update:** merge training and selection data from the previous time step $t - 1$ and the new data samples into $\mathcal{D}^t_{\text{train}}$ and $\mathcal{D}^t_{\text{selection}}$
- **(d) Model update:** re-estimate the parameters inside the neural QA based on $\mathcal{D}^t_{\text{train}}$ and $\mathcal{D}^t_{\text{selection}}$ at time t
- **(e) Learning effect:** evaluate the new QA model on $\mathcal{D}_{\text{learn}}$
- **(f) Robustness against forgetting:** evaluate the new QA model on a hold-out fraction $\mathcal{D}_{\text{forgetting}}$ of the SQuAD corpus

The above procedure allows us to study the overall effect of continuous learning on $\mathcal{D}_{\text{learn}}$, as well as the robustness against forgetting on the SQuAD dataset $\mathcal{D}_{\text{forgetting}}$. We again point out that all evaluations in the following are conducted on hold-out data that has been used neither for training, simulating users nor model selection.

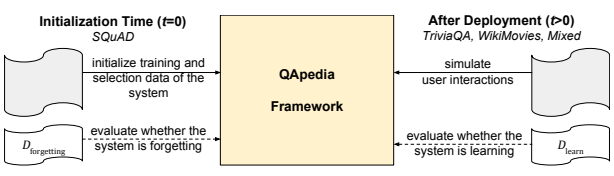

Figure 3: Overview of different corpora used to initialize our QApedia framework, as well as to simulate users.

[3] We used the preprocessed dump from https://github.com/facebookresearch/DrQA.

5.4 Clairvoyant, Noisy, and Adversarial Feedback

We study QApedia under different types of user responses:

(1) Clairvoyant users always provide feedback that is accurate. Hence, these users respond to a correct answer from the QA system with an up-vote and, conversely, with a down-vote if it is wrong. These users underlie our initial experiments in order to confirm continuous learning, while subsequent experiments control for the possibility of errors in the user feedback.
(2) Noisy feedback tests the system when a certain portion of user responses deviate from the ground-truth. Here we assume that a certain share of users (i. e., experts) respond with the correct answer, while others provide random feedback with probability ε.
(3) Adversarial feedback refers to feedback that is intentionally incorrect; i. e., up-votes if an answer is wrong and down-votes if an answer is correct. Here we again assume users that follow the ground truth and others that respond adversarially with probability ε.

For noisy and adversarial users, the total proportion of incorrect up-votes is not only determined by ε, but also by the performance of the system itself. For instance, if a system answers 20 % of the questions correctly and 10 % of the users vote randomly whereas 90 % provide correct feedback, then approximately one out three up-votes will be noisy. In our experiments, we chose a system-specific initialization of $\varepsilon = 0.05$, so that 18 % of the initial up-votes are noisy when averaged over all users. Due to its probabilistic nature, this number can vary across individual batches of user simulation and reach, e. g., up to 30 % in some of them.

Users are allowed to provide negative feedback repeatedly. Let ρ refer to the maximum frequency with which negative feedback is provided. In our experiments, we set $\rho = 1$ in order to yield a challenging setup and later experiment with $\rho = 3$ as part of a sensitivity analysis.

5.5 Implementation Details

As part of the answer ranking (P3), we train deep neural networks using mini-batch gradient descent over the content repository, i. e., the question-answer-paragraph triples, with a batch size of 32. We implemented a loss function $\mathcal{L}$ following [11]. Let a^* refer to the ground-truth answer, which spans from a_s^* to a_e^*. Given predictions $\boldsymbol{p}^s$ and $\boldsymbol{p}^e$ from the neural network, we then minimize the negative log-likelihood

$$\mathcal{L} = -\log \frac{\exp(p^s_{a^*_s})}{\sum_{j=1}^{l} \exp(p^s_j)} - \log \frac{\exp(p^e_{a^*_e})}{\sum_{j=1}^{l} \exp(p^e_j)}. \tag{10}$$

For this purpose, we utilize the AdaMax optimizer [22] with gradient clipping set to 10. All LSTM layers entail 128 hidden units and are trained with a recurrent dropout of 0.4. For all encoding layers, we use three BiLSTM layers, e. g., $m = 2$.

6 RESULTS

We conducted more than 2,500 hours of experiments and report results over more than 1.5 million simulated user interactions with 150 model re-estimations. We divide the results into three sections. First, we illustrate the learning process given shallow user feedback. Second, we check the robustness of our approach against forgetting. Third, we simulate noisy and adversarial users and evaluate whether our system still manages to improve under these challenging conditions.

6.1 Performance Improvements from Shallow User Feedback

The objective behind our first set of experiments is to determine the overall performance improvements from shallow user feedback. For this reason, we study the influence of user feedback when users always provide the correct response and, hence, consider results in the absence of noise or errors. This type of user interactions enables us to obtain an upper bound to the performance.

Fig. 4 reports our key results. In detail, we measure the fraction of correctly answered questions, e. g., where the predicted answer matches exactly to one of the ground truth answers. Per design, a traditional QA system fails to incorporate user feedback and is thus limited to a static performance. By contrast, our plots consistently derive considerable performance improvements from user feedback. Furthermore, we compare QApedia to a baseline without distant supervision in which we append samples with positive feedback directly into the training set.

- **WikiMovies:** Prior to the inclusion of user feedback, the systems obtain 13.3 % exact matches, which is identical to the performance of a static QA system. Our learning framework successfully manages to improve the performance over time. Distant supervision with shallow feedback increases the share of exact matches up to 35.5 % – an increase of 22.2 percentage points. On the other hand, the exact matches from the baseline without distant supervision amount to 30.9 %, which is inferior to QApedia by 4.6 percentage points. As determined later, the difference between this baseline and the performance of our system is attributable to the quality improvement of samples during the distantly supervised data generation process.
- **TriviaQA:** Distant supervision with $\rho = 3$ yields the best performance overall. Choosing $\rho = 1$ at first resembles the baseline without distant supervision in terms of performance but, after sufficient user interactions, outperforms it.
- **Combined dataset:** The combined dataset shows similar results. The performance differences are small, and yet the QApedia with $\rho = 3$ is still ranked first.

Evidently, continuous learning can be efficiently achieved through shallow user feedback using only a fraction of the original dataset. Choosing a larger τ yields much richer information, and yet both variants show little difference in performance, suggesting that simple feedback is already sufficient.

6.2 Robustness Against Forgetting in Neural QA

Our neural QA system could theoretically – as is the case for all other neural networks – be prone to catastrophic forgetting and we thus evaluate whether this issue is attenuated by our design. Fig. 5 compares the performance across user interactions: we measure the robustness of how well a QA system can answer questions regarding the initialization corpus (i. e., its hold-out part $\mathcal{D}_{\text{forgetting}}$) over time

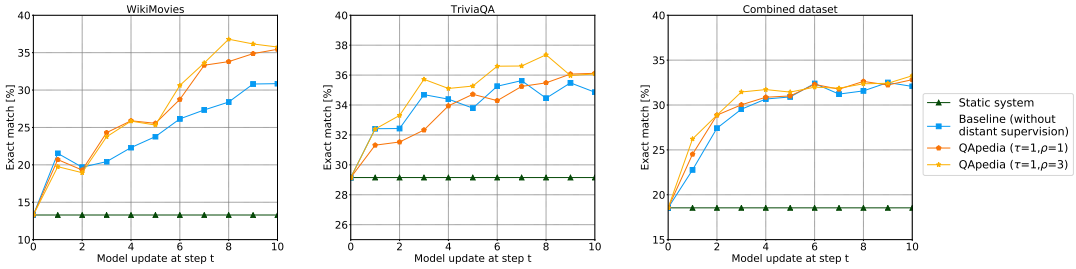

Figure 4: In a clairvoyant setting where correct feedback is provided, our QApedia achieves considerable performance improvements of up to 20 percentage points in terms of exact matches. The primary baseline represents the static QA system (i. e., horizontal line). For comparison, a trivial update framework without distant supervision is also reported.

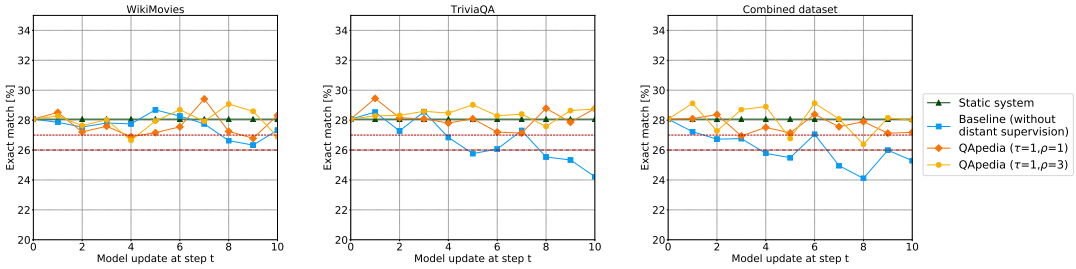

Figure 5: In a clairvoyant setting where only correct feedback is provided, our QApedia is considerably less prone to forgetting than alternative baselines. For better readability, a threshold is plotted at a 2-percentage-point decrease in performance in order to aid comparisons.

as user feedback is included. Small fluctuations are natural due to the data-driven learning method.

- **WikiMovies:** All system configurations reveal fairly similar patterns. Overall, our QApedia framework performs best with distant supervision and $\rho = 3$ repeated answers. This configuration not only shows continuous improvements for the feedback dataset, but only once attains a performance decrease of more than 1 % relative to the initial system and, moreover, reveals a gradual upward trend. The baseline without distant supervision performs less stably in this setting but, contrary to the next two datasets, does not show forgetting.
- **TriviaQA:** The performance of the baseline without distant supervision declines dramatically: the system's performance drops from 28.1 % exact matches down to only 24.2 %. In contrast, distant supervision again obtains a fairly stable performance.
- **Combined dataset:** The findings point in the expected directions, and yet we see that the mixed original of question-answer pairs in this dataset presents a challenging undertaking and results in considerable volatility. Again, the baseline without supervision hints at structural patterns of forgetting, as it regularly falls short by 2 % or more. Both distantly supervised approaches behave similarly and never undercut the 2 % mark.

Overall, our approach of distant supervision, and especially utilizing a larger value of ρ (i. e., 3 as opposed to 1), can reduce the risk of forgetting. Since the user feedback was simulated correctly (without noise), we credit this for the improved data generation via distant supervision. To obtain a better understanding, we quantitatively analyzed 200 question-answer-paragraph triplets generated from positive feedback by QApedia and the baseline without distant supervision. In the clairvoyant setting, all answers are correct, and yet we still see a distinct difference in the quality of the paragraphs. We found that, for QApedia, only about 4 % of data samples were uninformative (i. e., although the answer is correct contained within the paragraph, the paragraph alone is not sufficient to answer the question). For the baseline without distant supervision, the fraction of uninformative samples was considerably higher and amounted to 13 %.

6.3 Performance Under Noisy User Feedback

Being robust to user feedback with noise is a key requirement for use cases on the web. This experiment thus studies the effect of noise on the overall performance with noisy users as described in Section 5.4. We fix ρ to 1, and vary τ between 1 and 2. As described above, τ controls how rigorously we filter feedback during the credibility validation.

We report the experiment based on the WikiMovies dataset in Fig. 6. The total improvement declines only slightly as a result of the presence of noisy users: while it was up to 22.2 percentage points in the previous experiment based on feedback that was always correct, we now still achieve an increase in exact matches by 20.6 percentage points ($\tau = 2$) and 18.1 percentage points ($\tau = 1$). More importantly, the baseline (where positive feedback is directly added to the training set) sees an improvement of merely 9.8 percentage points, which is less than half of the performance gain from our QApedia framework. We further see that the QApedia

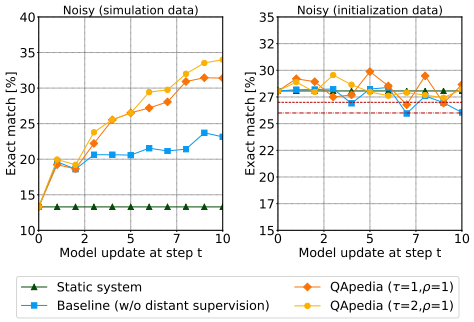

Figure 6: The proposed QApedia framework addresses a web-based use case where user feedback can be subject to noise, but remains fairly robust. Reported is the WikiMovies dataset.

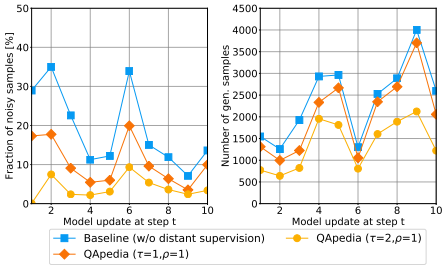

Figure 7: Our QApedia framework with automated credibility check for answers is considerably more robust than the baseline without distant supervision: our approach for the WikiMovies dataset removes more samples that received incorrect feedback (left) and thus results in fewer such samples that are added to the training set (right). This behavior is reinforced when choosing a stricter credibility check (e. g., $\tau = 2$).

framework, especially with stricter filtering, is also less prone to forgetting as compared to the baseline.

We now provide descriptive analysis concerning the update mechanism. We report (see Fig. 7 left) the relative frequency of incorrect samples that are added to the training data $\mathcal{D}_{\text{train}}$. Here we find that the baseline without distant supervision generates new training samples of which, on average, 19,2 % are wrong. Distant supervision decreases this share significantly. It plummets to 10.5 % for distant supervision with $\tau = 1$ and to 3.5 % for $\tau = 2$. Fig. 7 (right) reports the number of total samples that are generated (correct and incorrect). Here a higher τ decreases the number of samples added to the training set, but also results in fewer samples with incorrect feedback.

6.4 Robustness to Adversarial User Feedback

We repeat the previous experiments with users among whom an average of 20 % of the feedback is adversarial, i. e., users up-vote wrong answers and down-vote correct answers. The performance is depicted in Fig. 8 and resembles the previous findings under noise. The ratio of exact matches in our proposed approach increases by

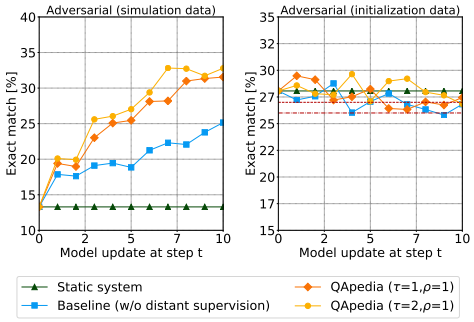

Figure 8: The proposed QApedia framework addresses a web-based use case where user feedback can be subject to adversarial inputs, but remains fairly robust.

19.4 percentage points, which is significantly larger than for the baseline without distant supervision (i. e., 11.8-percentage-point gain). Adversarial feedback leads to a larger variation in performance of the initialization task, where QApedia (in contrast to the baseline) again registers a better mean value. This is an interesting observation, since individual re-estimation steps can entail feedback of which 30 % is wrong. Altogether, these findings concerning adversarial feedback highlight the robustness of QApedia, especially with more rigorous credibility validation ($\tau = 2$).

7 DISCUSSION

This work presents QApedia: the first question-answering system for encyclopedic content repositories that gradually improves over time by harnessing shallow user feedback. A careful design was necessary in order to obtain state-of-the-art performance and, moreover, tailor the framework to the specific requirements of on-line feedback mechanisms on the web. We identified the following levers as especially pertinent. First, continuous learning from feedback can bolster the performance considerably; e. g., the proportion of exact matches can rise by up to 20 percentage points. We could have theoretically built our system such that it asks users to provide the correct labels as feedback (i. e., active crowd sourcing); however, we specifically decided upon shallow binary feedback, since, from a user perspective, this is considerably less costly. Second, our results establish that the problem of catastrophic forgetting can be effectively solved. Here it is important that all parts can efficiently deal with training sets that are dynamic in size and over time. For this reason, we utilize document retrieval, which adapts to the corpus size, and implement a novel distant supervision based on asynchronous updates. Third, we present a demanding setup with extensive adversarial feedback, and yet it does not necessarily diminish performance. Rather, our framework yields improvements of up to 30 %. This highlights the relevance of a simple credibility check.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments. We appreciate the help of Ryan Grabowski in editing our manuscript with regard to language. We gratefully acknowledge the support of NVIDIA Corporation, who donated of the TITAN Xp GPUs used for this research.

REFERENCES

[1] Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2018. Never-Ending Learning for Open-Domain Question Answering over Knowledge Bases. In *The Web Conference (WWW)*. 1053–1062.

[2] Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated Template Generation for Question Answering over Knowledge Graphs. In *International Conference on World Wide Web (WWW)*. 1191–1200.

[3] David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Müller, Maarten de Rijke, Stefan Schlobach, M. Voorhees, and L. Buckland. 2004. Using Wikipedia at the TREC QA Track. In *Text REtrieval Conference (TREC)*.

[4] Muhammad Asaduzzaman, Ahmed Shah Mashiyat, Chanchal K. Roy, and Kevin A. Schneider. 2013. Answering questions about unanswered questions of stack overflow. In *Working Conference on Mining Software Repositories*. 97–100.

[5] Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. 2017. Learning from User Interactions in Personal Search via Attribute Parameterization. In *ACM International Conference on Web Search and Data Mining (WSDM)*. 791–799.

[6] 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)*. 1533–1544.

[7] Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha. 2008. Finding the right facts in the crowd: factoid question answering over social media. In *International Conference on World Wide Web (WWW)*. 467.

[8] Davide Buscaldi and Paolo Rosso. 2006. Mining knowledge from Wikipedia for the question answering task. In *International Conference on Language Resources and Evaluation (LREC)*.

[9] YongGang Cao, Feifan Liu, Pippa Simpson, Lamont Antieau, Andrew Bennett, James J. Cimino, John Ely, and Hong Yu. 2011. AskHERMES: An online question answering system for complex clinical questions. *Journal of Biomedical Informatics* 44, 2 (2011), 277–288.

[10] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, JR., and Tom M. Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*.

[11] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Association for Computational Linguistics (ACL)*. 1870–1879.

[12] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Seán Slattery. 2000. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence* 118, 1-2 (2000), 69–113.

[13] Abdessamad Echihabi, Ulf Hermjakob, Eduard Hovy, Daniel Marcu, Eric Melz, and Deepak Ravichandran. 2008. How To Select An Answer String? In *Advances in Open Domain Question Answering*. 383–406.

[14] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 1156–1165.

[15] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building Watson: An Overview of the DeepQA Project. *AIMag (AI Magazine)* 31, 3 (2010), 59.

[16] Robert L. Grossman (Ed.). 2005. *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*.

[17] Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2000. FALCON: Boosting Knowledge for Answer Engines. 479–488.

[18] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*. 1693–1701.

[19] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. In *Association for Computational Linguistics (ACL)*. 963–973.

[20] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Association for Computational Linguistics (ACL)*. 1601–1611.

[21] Michael Kaisser and Tilman Becker. 2004. Question Answering by Searching Large Corpora with Linguistic Methods. In *Text REtrieval Conference*.

[22] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[23] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America* 114, 13 (2017), 3521–3526.

[24] Bernhard Kratzwald and Stefan Feuerriegel. 2018. Adaptive Document Retrieval for Deep Question Answering. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[25] Bernhard Kratzwald and Stefan Feuerriegel. 2019. Putting Question-Answering Systems into Practice: Transfer Learning for Efficient Domain Customization. In press. (2019).

[26] Eun-Ju Lee and Yoon Jae Jang. 2010. What Do Others' Reactions to News on Internet Portal Sites Tell Us? Effects of Presentation Format and Readers' Need for Cognition on Reality Perception. *Communication Research* 37, 6 (2010), 825–846.

[27] Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* (2016).

[28] Liangda Li, Hongbo Deng, Anlei Dong, Yi Chang, Ricardo Baeza-Yates, and Hongyuan Zha. 2017. Exploring Query Auto-Completion and Click Logs for Contextual-Aware Web Search and Query Suggestion. In *International Conference on World Wide Web (WWW)*. 539–548.

[29] Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Association for Computational Linguistics (ACL)*. 1736–1745.

[30] 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)*. 1400–1409.

[31] T. Mitchell, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, W. Cohen, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, J. Welling, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, and M. Gardner. 2018. Never-ending learning. *Commun. ACM* 61, 5 (2018), 103–115.

[32] Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. 2005. Probabilistic question answering on the web. *Journal of the Association for Information Science and Technology* 56, 6 (2005), 571–583.

[33] 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)*. 2383–2392.

[34] Gerard Salton and Chris Buckley. 1990. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science* 41, 4 (1990), 288–297.

[35] Avneesh Saluja, Ian Lane, and Ying Zhang. 2012. Machine translation with binary feedback: a large-margin approach. In *Proceedings of the Tenth Biennial Conference of the Association for Machine Translation in the Americas*.

[36] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* (2016).

[37] Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a Conversational Agent Overnight with Dialogue Self-Play. *arXiv preprint arXiv:1801.04871* (2018).

[38] Dan Shen and Dietrich Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *Annual Meeting of the Association for Computational Linguistics*. 889–896.

[39] R. F. Simmons. 1965. Answering English questions by computer: A survey. *Commun. ACM* 8, 1 (1965), 53–70.

[40] Yu Su, Shengqi Yang, Huan Sun, Mudhakar Srivatsa, Sue Kase, Michelle Vanni, and Xifeng Yan. 2015. Exploiting Relevance Feedback in Knowledge Graph Search. In *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 1135–1144.

[41] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. *Neural Information Processing Systems (NIPS)* (2015), 2440–2448.

[42] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over RDF data. In *Conference on World Wide Web (WWW)*.

[43] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced Ranker-Reader for Open-Domain Question Answering. In *Conference on Artificial Intelligence (AAAI)*.

[44] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated Self-Matching Networks for Reading Comprehension and Question Answering. In *Annual Meeting of the Association for Computational Linguistics*. 189–198.

[45] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *International Conference on Machine Learning (ICML)*.

[46] Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *ACM International Conference on Information & Knowledge Management (CIKM)*. 1107–1116.