Never-Ending Learning for Open-Domain Question Answering over Knowledge Bases

Abdalghani Abujabal
Max Planck Institute for Informatics
Saarland Informatics Campus, Germany
abujabal@mpi-inf.mpg.de

Rishiraj Saha Roy
Max Planck Institute for Informatics
Saarland Informatics Campus, Germany
rishiraj@mpi-inf.mpg.de

Mohamed Yahya
Bloomberg L.P.
London, United Kingdom
myahya@myahya.org

Gerhard Weikum
Max Planck Institute for Informatics
Saarland Informatics Campus, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

Translating natural language questions to semantic representations such as SPARQL is a core challenge in open-domain question answering over knowledge bases (KB-QA). Existing methods rely on a clear separation between an offline training phase, where a model is learned, and an online phase where this model is deployed. Two major shortcomings of such methods are that (i) they require access to a large annotated training set that is not always readily available and (ii) they fail on questions from before-unseen domains. To overcome these limitations, this paper presents NEQA, a *continuous learning paradigm* for KB-QA. Offline, NEQA automatically learns templates mapping syntactic structures to semantic ones from a *small* number of training question-answer pairs. Once deployed, continuous learning is triggered on cases where templates are insufficient. Using a semantic similarity function between questions and by judicious invocation of non-expert user feedback, NEQA learns new templates that capture previously-unseen syntactic structures. This way, NEQA gradually extends its template repository. NEQA periodically re-trains its underlying models, allowing it to adapt to the language used after deployment. Our experiments demonstrate NEQA's viability, with steady improvement in answering quality over time, and the ability to answer questions from new domains.

KEYWORDS

Question answering; Never-ending learning; User feedback

ACM Reference Format:
Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2018. Never-Ending Learning for Open-Domain Question Answering over Knowledge Bases. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186004

1 INTRODUCTION

Motivation. Open-domain question answering over knowledge bases (KB-QA) is an active research area where the goal is to provide crisp answers to natural language questions [1, 6, 9, 24, 42, 56, 57, 62] or telegraphic queries [31, 43]. An important direction in KB-QA performs this answering via semantic parsing: translating a user's *question* to a SPARQL *query* that is subsequently executed over a KB like Freebase [12], DBPedia [21] or YAGO [45]. Existing approaches rely on a clear separation between an offline training phase, where a model is either learned or manually crafted, and an online phase where this model is deployed to answer users' questions. Such approaches suffer from three major shortcomings: (i) they require access to reasonably large training sets with sufficient syntactic and lexical coverage representative of the kinds of questions users pose, which are expensive to construct, (ii) they provide no mechanism for improving their performance over time by learning from failure cases on questions received after deployment, and (iii) they are limited to the language learned at training time, therefore, they fail on questions from domains not observed previously.

In this work, we present a continuous-learning framework for template-based KB-QA called NEQA (*N*ever *E*nding *QA*) that (i) is initialized with a *small* training set, (ii) improves its performance over time by judiciously invoking user feedback on answers from

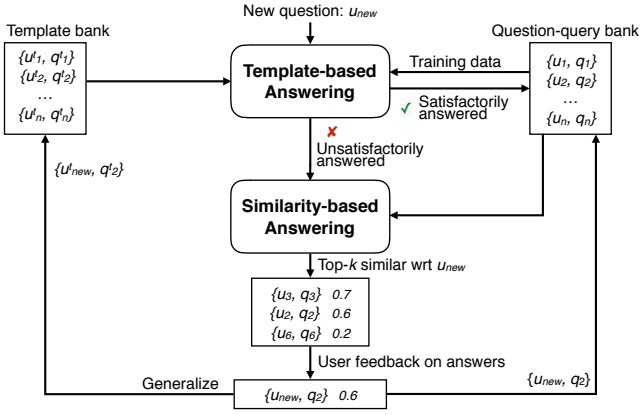


Figure 1: Continuous learning: if a new question u_{new} cannot be satisfactorily answered via templates, we utilize user feedback on the output of a semantic similarity function to learn a new template (u^t_{new}, q^t) based on u_{new}.

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186004

non-expert users on the failure cases of the underlying template-based answering mechanism, and (iii) adapts to the language used after deployment by periodically retraining its underlying models. A simplified workflow is shown in Figure 1.

Training a well-performing open-domain KB-QA system requires a massive annotation effort, in terms of cost, time and expertise. Some methods use labeled SPARQL queries [16], while others train their systems on question/answer pairs as a form of weak supervision [6, 9], which has been proven to work well. We adopt this form of supervision, however, for only a small training seed to minimize the annotation effort required. We rely on non-expert user feedback to acquire more question/answer pairs over time. In our experiments, we show that NEQA was able to successfully answer questions from domains it has not seen before.

We harness non-expert user feedback on answer sets generated as a response to a given question, which is related to a number of recent ideas in semantic parsing and natural language interfaces to databases (NLIDB). Li et al. [35] invoke user feedback to resolve ambiguous words/phrases in the users' questions, while Iyer et al. [28] ask expert users to provide a full SQL query that answers a question over a database. In Wang et al. [51], an end user teaches the model new concepts through direct interaction. Other approaches utilize crowdsourcing as a sort of interactivity [52, 53].

NEQA builds on an established line of work that performs semantic parsing by translating syntactic dependency structures of utterances to semantic predicate-argument structures using templates that are either manually crafted [23, 24, 40, 48, 57], or automatically learned [1]. By exploiting syntax, such template-based approaches achieve better generalization [7]. As an example, such systems can use a template generated from u_1 = *"which film awards was bill carraro nominated for?"* to answer the syntactically isomorphic question *"which president was lincoln succeeded by?"*, despite the fact that it invokes a different semantic KB predicate.

The main drawback of such systems is their inability to handle new syntactic structures beyond those observed in the static training set. Take, for example, a new question u_{new} = *"what are the film award nominations that bill carraro received?"*. Even if the above systems had seen u_1 during training, they cannot answer this new semantically related (but syntactically different) question. This problem is exacerbated if these systems are trained on a small number of training examples. NEQA rectifies this limitation by using a state-of-art similarity function [63] to find a correctly answered and semantically-similar question from its history, and subsequently learns a new template based on the new question.

Approach. NEQA is driven by two intuitions. First, syntactic isomorphism of questions is a strong cue for the isomorphism of their respective predicate-argument structures (SPARQL queries). This intuition underlies template-based approaches outlined above. Second, where syntactic isomorphism fails, NEQA invokes a semantic similarity function together with user feedback to transfer semantics across syntactic structures and triggers the learning of new templates. NEQA combines these intuitions into a *continuous learning framework* that gradually overcomes the limitations of small training sets and evolves over time.

NEQA starts by automatically learning a few templates from a small number of questions offline, using the approach of Abujabal et al. [1]. Figure 1 shows what happens when NEQA receives a new question u_{new} online. It adds satisfactorily answered questions to an ever-growing bank of question-query (u, q) pairs, initially composed of a small training set. Questions in this bank will be called upon when our template-based answering mechanism fails to satisfactorily answer a new question. In such cases, NEQA learns new syntactic structures to improve its future answering performance.

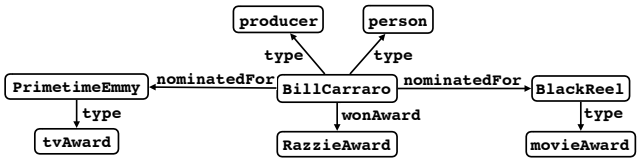


Figure 2: A subgraph from a knowledge base. Nodes correspond to entities and classes, and edges represent predicates.

When u_{new} is unsatisfactorily answered using our template-based answering mechanism, NEQA triggers the learning of a new template from this question. It first consults a semantic similarity function to find the k previously answered questions closest to u_{new}. NEQA then instantiates the corresponding queries with entities from u_{new}. Leveraging user feedback on answer sets generated by executing these queries over the KB, one of the resulting queries (q_2 in Figure 1) is determined to be the best fit for u_{new}. NEQA then uses a lexicon and an Integer Linear Program to align the constituents of u_{new} and q_2. A new template (u^t_{new}, q^t_2) is created from this pair, which is then added to the template bank.

Contributions. We present NEQA, the first continuous learning framework for KB-QA, and make four novel contributions:

- a KB-QA system that can be seeded with a small number of training examples and supports continuous learning to improve its answering performance over time;
- a similarity function-based answering mechanism that enables NEQA to answer questions with previously-unseen syntactic structures, thereby extending its coverage;
- a user feedback component that judiciously asks non-expert users to select satisfactory answers, thus closing the loop between users and the system and enabling continuous learning;
- extensive experimental results on two benchmarks demonstrating the viability of our continuous learning approach, and the ability to answer questions from previously-unseen domains.

2 SETUP

2.1 Basic Concepts

Knowledge base. A knowledge base (KB) is a collection of facts represented as a graph where nodes correspond to (i) entities $e \in E$ (e.g., `BillCarraro`), (ii) classes or types $c \in C$, (e.g., `movieAward`), and (iii) literals $h \in H$ (e.g., dates). Two nodes are connected by an edge labeled with a predicate $p \in P$ (for example, `nominatedFor`), which together form a triple (e.g., `BillCarraro nominatedFor BlackReel`). Any $s \in S = E \cup C \cup P$ is called a *semantic item*. Figure 2 shows a sample KB subgraph.

Query. To query a KB, we use graph-matching based on SPARQL triple patterns. A triple pattern is a triple with one or more of its components replaced by variables (e.g., `?x type movieAward`). A query

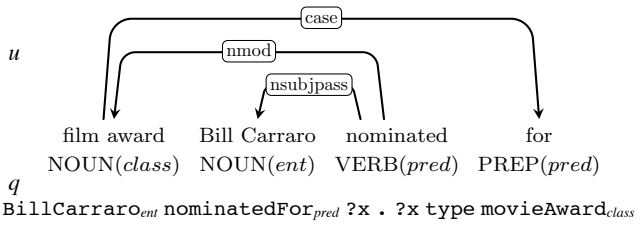


Figure 3: An aligned question-query pair (u, q). Alignment is indicated by shared *ent*, *pred*, and *class* annotations.

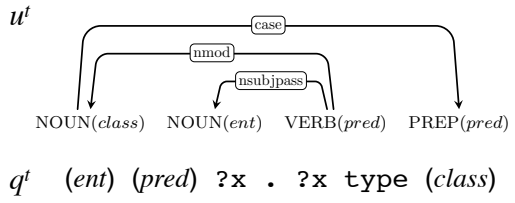


Figure 4: A template composed of aligned question and query templates (u^t, q^t) generated from the question-query pair in Figure 3. Shared *ent*, *pred*, and *class* annotations indicate alignment between u^t and q^t.

q is a set of triple patterns (e.g., `BillCarraro nominatedFor ?x . ?x type movieAward`). The variable `?x` is designated as the *projection variable*.

Answer. An answer a to a query q over a KB is an entity (possibly a set) which is obtained by mapping variables of q to KB items where the projection variable maps to a. For example, the answer to the above query is `BlackReel` (Figure 2).

2.2 Question and Query Templates

Templates play an important role in KB-QA [1, 24, 40, 48]. They guide the mapping of the syntactic structures of natural language utterances to semantic predicate-argument structures in SPARQL queries. Figure 4 shows an example template from Abujabal et al. [1], which has the same form as the templates we use in this work. It consists of a question template u^t and its corresponding query template q^t, where u^t and q^t are derived from generalizations over the dependency parse and the query, respectively. Alignment of the constituents of u^t and q^t is indicated by shared *ent*, *pred*, and *class* annotations.

Generating templates. We follow the approach of Abujabal et al. (Section 3 in [1]) for learning templates. We briefly summarize the relevant parts of the method here for completeness. The approach is designed for a weakly supervised setting where a training instance is a question u paired with its answer set A_u [9]. NEQA uses this form of supervision for the initial training phase. The approach heuristically generates a query q that captures each training question u from the corresponding training pair (u, A_u). For $u =$ *"Which film awards was Bill Carraro nominated for?"*, the corresponding query would be $q =$ `BillCarraro nominatedFor ?x . ?x type movieAward`. We now have a question query pair (u, q). The rest of the discussion explains how a template is generated from such a pair. This process is invoked in NEQA both as part of initial training (where we start with (u, A_u) pairs), and during continuous learning, where NEQA generalizes a (u, q) pair resulting from the similarity function and user feedback to a template (Figure 1).

Next, nodes in the dependency parse of u are aligned with semantic items in q. A dependency parse is a tree whose nodes correspond to words in a sentence and edges represent grammatical relations between words. We use the Stanford dependency parser [17] in this work. For example, *'film awards'* in u above is aligned with the KB class `movieAward` in the corresponding q. A weighted *lexicon* L (Section 2.3) is used to connect phrases in u to all candidate semantic items in q, forming a weighted bipartite graph. The alignment problem is formulated as a constrained optimization problem solved using an Integer Linear Program (ILP) [1]. The solution to the ILP is a role-aligned question-query pair (Figure 3) where phrases in u that are not part of the alignment are dropped (e.g., *'Which'*). Finally, concrete values in both u and q are dropped to produce a role-aligned question-query template pair (u^t, q^t) (Figure 4).

Using templates. During answering, when a new question u_{new} is encountered, question templates matching its dependency parse are identified (see Section 3.2). Corresponding query templates are then instantiated using alignment information and the lexicon. This step potentially generates multiple query candidates due to lexicon ambiguity, which are ranked using a learning-to-rank (LTR) model. Finally, the answer of the top-ranked query is presented to the user.

2.3 Predicate and Class Lexicons

To connect utterance vocabulary to semantic items in the KB, we construct a lexicon L that consists of a predicate lexicon L_P and a class lexicon L_C. L_P and L_C are constructed from ClueWeb09-FACC1, a corpus of $500M$ Web pages annotated with Freebase entities [25]. To construct L_P we run the extraction pattern *"e_1 r e_2"* over the corpus, where e_1 and e_2 are entities and r is a phrase. Following the *distant supervision hypothesis* in Mintz et al. [38], we assume that if $(e_1\ p\ e_2)$ is a triple in the KB, then r expresses p and we add $r \mapsto p$ to L_P. L_C is constructed by running Hearst patterns [27] over the corpus, where one argument is an entity and the other is a noun phrase. For example, for *"e and other np"*, the entry $np \mapsto c$ for each c is added to L_C such that $(e$ `type` $c) \in KB$. Each entry in L is assigned a mapping weight proportional to its corpus frequency. For handling entities in questions, we use an existing named entity recognition and disambiguation (NERD) system [59].

3 THE NEQA FRAMEWORK

Initially, NEQA goes through an offline training stage that populates its question-query and template banks with their seed data (Section 3.1). Online, when NEQA is deployed, a stream of questions arrives from users. NEQA attempts to answer each incoming question using the templates it has learned so far (Section 3.2). If this fails, it falls back to answering using the semantic similarity function against the set of already answered questions in the question-query bank (Section 3.3). In both cases, NEQA utilizes user feedback on answer sets to extend its banks (Section 3.4). After each *batch* of questions, NEQA retrains its learning-to-rank (LTR) ranking model on the accumulated data in its banks to improve system performance for subsequent questions.

3.1 Initial Training

NEQA is initialized through an automated template generation stage, as discussed in Section 2.2. This stage relies on weak supervision through a small number of questions, each paired with its answer set. This training stage results in populating the question-query and template banks (Figure 1) with their seed data that are used for bootstrapping the continuous learning process. Moreover, it results in NEQA's first LTR model. NEQA's continuous learning improves all three components once the system goes online.

Questions in the question-query bank are stored in a generalized form that facilitates improved matching by the semantic similarity function. Specifically, entities in both questions and queries in the question-query bank are replaced by placeholders. For example, *"Which film award was ENTITY nominated for?"* (question) is paired with `ENTITY nominatedFor ?x . ?x type movieAward` (query).

3.2 Answering with Templates

Once the system goes online, it starts receiving new question utterances from users. Given a new question u_{new}, NEQA identifies matching question templates $\{u^t\}$ in its template bank. A match is deemed successful if edge labels and POS tags in the dependency parse of u_{new} and u^t agree. For example, the dependency parse of *"which president was lincoln succeeded by?"* matches the utterance template in Figure 4. When this happens, the associated query template q^t is instantiated with concrete semantic items using the phrases in u_{new}, the alignment information between u^t and q^t, and the underlying lexicon L. For example, based on the alignment information in Figure 4, the verb phrase *'succeeded by'* is used to instantiate a KB predicate.

Note that a single utterance may match multiple utterance templates, and these templates may result in multiple queries due to ambiguity in L. The learning-to-rank (LTR) model is used to rank this set of candidate queries. Features for training the LTR model are borrowed from past work [1, 6], and are derived from lexicon weights, entity popularity scores, answer type constraints, and sizes of answer sets, among others. The top-ranked queries, as detailed below, are then executed over the KB to fetch answer sets.

Next, user feedback is obtained on these retrieved answer sets. To be realistic, note that we obtain feedback on answer sets of the top-k queries where k is small. If an answer set is chosen (e.g., `{AndrewJohnson}`) by the user, then this validates the choice of the query q^* that generated this answer set as correct (e.g., `AbrahamLincoln succeededBy ?x . ?x type president`). On the other hand, when none of the shown answer sets is chosen, NEQA proceeds differently (Section 3.3). Finally, the correct query q^* is paired with u_{new} and is then added to our question-query bank after entity generalization. For the example above, q^* after entity generalization is: `ENTITY succeededBy ?x . ?x type president`. Such an augmentation of the question-query bank potentially results in the system gaining questions with unseen KB predicates. We validate this postulate in our experiments on open-domain answering. When a batch of questions has been received, we use the questions answered satisfactorily by the templates to retrain the LTR model to further boost the performance of the system on subsequent questions.

3.3 Answering via Similarity Function

A core contribution of NEQA is to extend coverage of template-based answering using a semantic similarity function. A typical template-based KB-QA system fails when an input utterance represents previously unseen syntactic structure [1, 23, 48, 57]. Further, even when a matching utterance template is identified, the KB-QA system might fail to deliver answers due to errors in the alignment information between the question and the query templates.

NEQA, on the other hand, builds on failure cases to improve its future QA performance. Whenever a question cannot be answered satisfactorily using templates, NEQA uses a *semantic similarity function* to retrieve the k most semantically similar questions to u_{new} from its question-query bank. For example, say, the utterance u_{new} = *"what are the film award nominations that bill carraro received?"* represents a syntactic structure beyond the coverage of our current templates. However, our question-query bank contains a similar question: *"which film awards was bill carraro nominated for?"*. The goal of our similarity function is to identify such questions and allow the *transfer of semantics* across syntactic structures.

We first use an off-the-shelf NERD system to link mentions of entities in u_{new} to KB entities [59]. Identified entities in u_{new} are then replaced by placeholders to ensure better generalization. Similar generalization is also done on (u, q) pairs in our question-query bank (Section 3.1). Next, the corresponding queries $\{q_1 \dots q_k\}$ for these similar utterances are instantiated with entities from u_{new} and then executed over the KB to retrieve answer sets.

Next, we obtain user feedback on the answer sets of the k queries. If an answer set is chosen (e.g., `{BlackReel}`), the corresponding query q^* (e.g., `BillCarraro nominatedFor ?x . ?x type movieAward`) is paired with u_{new}. The newly generated pair (u_{new}, q^*) is then added to our question-query bank after entity generalization. A vital step of NEQA is the subsequent on-the-fly alignment and generalization of u_{new} and q^*, to obtain a new template (u^t, q^t). This is performed by casting the problem as an integer linear program (Section 2.2). The new template (u^t, q^t) is then added to NEQA's template bank. By acquiring more templates, the system's capability to handle syntactic variation increases, i.e., it learns how to *directly* answer questions with new syntactic structures.

Similarity function. Following recent work on *question retrieval* in community question answering [63], we opt for an unsupervised semantic similarity function. Note that we treat the similarity function as a plug-in, where supervised methods can also be used if required. Our similarity function consists of two components: (i) question likelihood based on a language model, and (ii) word embedding-based similarity obtained through *word2vec*.

Given a new question u_{new} and a question u_i from our question-query bank, our first component, based on language model, computes question likelihood as follows:

$$score_{LM}(u_{new}, u_i) = \prod_{w \in u_{new}} [(1-\lambda) \cdot P_{ml}(w|u_i) + \lambda \cdot P_{ml}(w|C)] \quad (1)$$

where $P_{ml}(w|u_i)$ represents the maximum likelihood probability estimate of w estimated from u_i and w is a unigram, bigram or trigram generated directly from u_{new} or from paths of lengths one and two in the dependency parse of u_{new}. $P_{ml}(w|C)$ is a smoothing term calculated as the maximum likelihood of w in a corpus C

of questions from our question-query bank, and $\lambda \in [0, 1]$ is a smoothing parameter.

The second component uses a *word2vec* model pre-trained on Google News corpora [37]:

$$score_{w2v}(u_{new}, u_i) = \frac{1}{|\mathcal{P}|} \sum_{(w_j, w_k) \in \mathcal{P}} cos(w2v(w_j), w2v(w_k)), \quad (2)$$

where, $w_j \in u_{new}$, $w_k \in u_i$, $w2v(w)$ is the *word2vec* embedding vector of w, and $\mathcal{P}$ is the set of word pairs from u_{new} and u_i whose cosine similarity is above a threshold τ.

The final score is a linear combination of the two components presented above, where α is a trade-off parameter:

$$\begin{aligned} score_{sim}(u_{new}, u_i) = \ & \alpha \cdot score_{LM}(u_{new}, u_i) \\ & + (1 - \alpha) \cdot score_{w2v}(u_{new}, u_i) \end{aligned} \quad (3)$$

3.4 Harnessing User Feedback

NEQA resorts to user feedback in two cases. The first is when an incoming utterance u_{new} is answered using templates in the template bank. In this case, the user is asked to give feedback on the relevance of the answer sets shown to her by either choosing the one that satisfies her information needs or none of them, if none is satisfactory. By propagating answer quality back to queries, this feedback is leveraged to extend the question-query bank. The second case is when NEQA returns answers using the semantic similarity function. The answers obtained from the top-k previously answered questions that are most similar to u_{new} are shown to the user for assessment. This feedback is used to extend both the template and the question-query banks.

In both cases above, it is important to keep k small to ensure the feasibility of asking for user feedback. In the experiments, we show that this is the case for our choices of LTR and semantic similarity functions. Additionally, we look at the extreme case where $k = 1$ and user feedback is bypassed by making the assumption that answers returned by our system are correct, and can be used for continuous learning.

4 EXPERIMENTS

We present extensive experimental evaluation and analysis of continuous learning in NEQA. Our experiments demonstrate NEQA's ability to continuously improve its answering performance over time starting with a very limited training set. We show that the answering performance of traditional state-of-the-art QA systems where periodic re-training is done is inferior to that of NEQA, which was designed specifically to support continuous learning. We also show that the manner in which NEQA exploits the interaction between syntax and semantics allows it to support truly open-domain QA by answering questions requiring predicates it has not seen before.

4.1 Setup

Benchmarks. We use the following KB-QA benchmarks over Freebase to evaluate NEQA:

- **WebQuestions (WQ)** [9]: This benchmark was created using Google's suggest API and crowdsourcing, and is composed of 5810 questions, each paired with its answer set. These are split into 3778 training and 2032 test instances.
- **ComplexQuestions (CQ)** [5]: This very recent benchmark focuses on more challenging multi-constraint questions. It contains 2100 question-answer pairs from (i) a commercial search engine, (ii) WebQuestions and, (iii) the benchmark released by Yin et al. [62]. It is split into 1300 training and 800 test cases.

Property	WQ	CQ
Size of initial training set	300	105
Size of development set	300	300
Initial templates acquired	223	85

Table 1: NEQA initialization statistics.

We base our extended analyses and comparisons with baseline systems on WQ due to the lack of publicly available KB-QA systems designed for handling complex questions in CQ. As detailed below, small subsets of the respective training sets are used for initial training, followed by streaming the complete test sets in batches to simulate online answering and continuous learning.

Training. Table 1 gives a summary of the initial training stage. A main motivation for resorting to continuous learning is the cost associated with obtaining a large training set upfront. To simulate small seed training sets, we randomly sample only about 8% of the standard WQ and CQ training sets. These seed training sets were used to initialize the (i) question-query and template banks (Section 3.1), (ii) learning-to-rank (LTR) models (Section 3.2), and (iii) language model component of the similarity function (Section 3.3). The development sets (randomly sampled from the training set) were used to tune the λ and α parameters of the similarity function. Crucially, NEQA is never exposed to the full WQ or CQ training sets in our experiments beyond the above seed examples. After initial training, NEQA is deployed to answer incoming questions, performing continuous learning when necessary.

Continuous learning. During answering, NEQA receives test questions from the respective benchmark in batches. At the end of each batch, we retrain the LTR component and re-estimate the language model with the data seen thus far. We set our batch size to 100 questions, so that we can observe the effect of continued learning over a larger number of batches (20 for WQ, 8 for CQ). Varying batch size did not have any significant effect on the observed trends.

We report system performance in two modes, which differ in their invocation of user feedback during continuous learning:

- **NEQA** (with user feedback): Here, we invoke user feedback to select the most appropriate answer set among the top-k answer sets obtained using templates and, if none are appropriate, among the top-k obtained using the similarity function. We use gold answer labels provided with the benchmarks to simulate user feedback. We use $k = 5$ in both cases, as we find that it provides a good balance between recall and the number of answer sets a user needs to look at.
- **NEQA-No-User-Feedback**: In this configuration, we perform continuous learning without user feedback. Instead, we take the top-ranked answer set in either of the two lists of answer sets

above to be the correct one. We consider the list of answer sets obtained using the similarity function only if the list obtained using templates (ranked by the LTR function) is empty. This configuration demonstrates the quality of our template-based and similarity function-based answering mechanisms and helps understand the gap filled by user feedback.

4.2 Results

Answering performance over time. Figure 5 shows how NEQA performs after deployment, as it receives user questions and invokes continuous learning where necessary. The cumulative average F1 scores in Figures 5a and 5d show an improvement over time for both benchmarks and for both feedback configurations. We compute the F1 score of a given batch after observing all questions in that batch, but before updating our ranking models based on that batch. The cumulative F1 score at batch n is the mean of these scores over all n batches. On both benchmarks, the general trend is for the cumulative F1 to increase as NEQA sees more questions and invokes continuous learning as needed to learn new templates and improve its ranking model. In general, the numbers on CQ are lower due to its more challenging nature stemming from multi-relation questions. We can see some fluctuation in the initial batches for both datasets. We attribute such variations to the modest ranking performance of the underlying LTR model during the very first iterations due to the small number of instances it was trained on. As expected, NEQA's F1 increases significantly when user feedback is invoked. We observe an F1 increase of 3.8 and 2.8 points over the no-feedback configuration for WQ and CQ, respectively.

Augmentation of banks. NEQA extends its banks with new templates and question-query pairs over time. Figures 5b and 5e show the number of templates learned over time. Each template captures a distinct syntactic structure and its mapping to the appropriate semantic predicate-argument structure. Each template in the template bank corresponds to one or more question-query pairs in the question-query bank: it was either generated from such a pair during training or continuous learning, or was used to answer u in that pair by mapping it to q. In general, having more correct (u, q) pairs in the question-query bank means: (i) NEQA has learned more correct templates, and, (ii) NEQA can better transfer these new templates to new syntactic structures with semantics similar to that of q. Figures 5c and 5f show the numbers of *correctly answered* new (u, q) pairs for WQ and CQ, respectively with the two modes of feedback. A (u, q) pair is deemed correct if the gold answer set of u overlaps with the answer set of q when executed over the KB. For 1393 out of 2032 test questions in WQ (338 out of 800 in CQ), user feedback indeed yielded the ground-truth answer set.

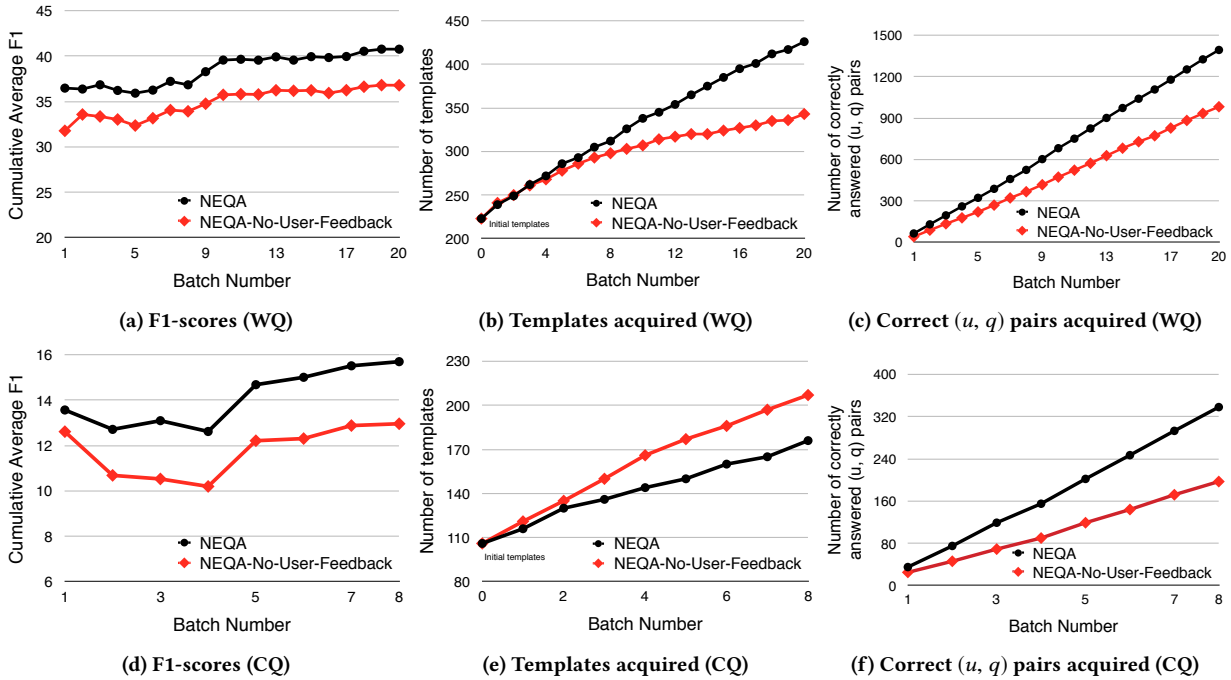
Contrasting these figures gives interesting insights. For WQ, the number of templates obtained from user feedback is higher than that obtained without. This is because the similarity function does a good job at surfacing the correct answer set to the top-k from which the user selects the correct one. However, without feedback, the top-1 may be incorrect, in which case alignment between the corresponding query and the question at hand fails (as opposed to generating a spurious alignment), resulting in no templates. The question-query bank in the configuration with feedback contains more correct (u, q) pairs, meaning that we have more correct alignments (and hence less spurious templates). When looking at CQ, we see that the no-feedback configuration results in more templates. The reason here is that with the complexity of the dataset and the limitations of the similarity function, users are more likely to decide that no answer set produced through the similarity function is appropriate. In the no-feedback case, the topmost answer set from the similarity function is always chosen, and alignments (including spurious ones) are more likely here due to the length of the questions. Despite the large number of templates for the no-feedback configuration, we can see that NEQA with user feedback results in more correct (u, q) pairs (Figure 5f), indicating that the no-feedback configuration has more spurious templates than the one with feedback, as expected.

Method	Avg. Prec.	Avg. Rec.	Avg. F1
QUINT [1] - No Feedback	25.5	30.2	25.7
QUINT [1] - Feedback	35.2	44.1	35.9
AQQU [6] - No Feedback	24.5	29.6	24.8
AQQU [6] - Feedback	36.3	45.2	37.6
NEQA-No-User-Feedback	36.6	45.4	37.0
NEQA	40.6	49.5	40.8

Table 2: Performance of continuous learning-based methods on the WebQuestions test set. User Feedback is used to retrain the systems after each batch.

Comparison with state-of-the-art. An intuitive baseline for evaluating continuous learning in NEQA is to extend existing static-learning based methods to our setting of continuous learning through user feedback and periodic retraining. Concretely, we trained both QUINT [1], and AQQU [6] on the same initial training seed as NEQA (300 question-answer pairs for WQ). Then, we streamed the 2032 test questions in batches of size 100, where user feedback is harnessed on the top-5 answer sets generated by the two systems. After each batch, the baseline systems were re-trained using the initial seed plus those questions from the batches seen thus far. Additionally, similar to NEQA-No-User-Feedback configuration, we performed continuous learning without user feedback for the two baselines.

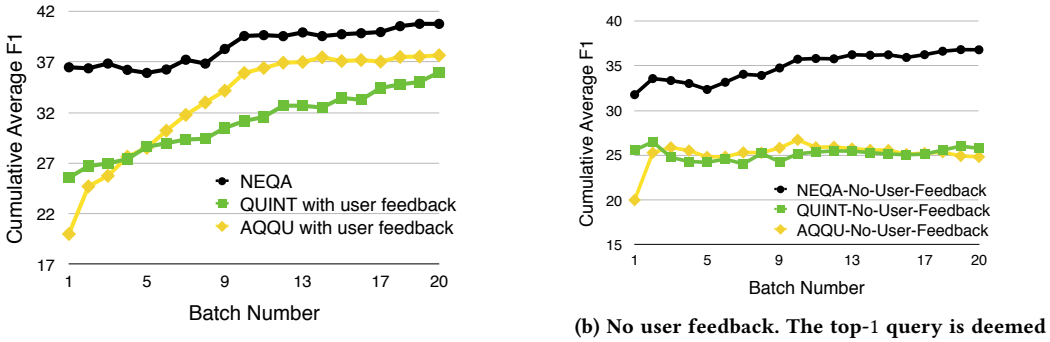
Aggregate results over the WQ test questions are shown in Table 2, where NEQA outperformed both QUINT and AQQU in the two modes of operation (with and without user feedback). To gain more insights, the per-batch cumulative average F1 for the three systems in both configurations is depicted in Figures 6a and 6b. NEQA starts off with a high F1 score (36.5) in batch one, compared to AQQU and QUINT with F1 scores of 20.0 and 25.5, respectively. This is due to two reasons: (i) NEQA learns templates online and adds them directly to the template bank, while QUINT, for example, learns new templates after each batch and (ii) when a question has no matching templates, our similarity function is invoked and might be able to correctly answer the question at hand, and hence, positively affects the performance. AQQU disregards the syntax of questions and relies on three query templates with exhaustive instantiation. This makes the underlying ranking module of AQQU



(a) F1-scores (WQ) (b) Templates acquired (WQ) (c) Correct (u, q) pairs acquired (WQ)

(d) F1-scores (CQ) (e) Templates acquired (CQ) (f) Correct (u, q) pairs acquired (CQ)

Figure 5: Performance of NEQA with and without user feedback as a function of batch number, on the WebQuestions (WQ) and ComplexQuestions (CQ) datasets.



(a) Harnessing user feedback.

(b) No user feedback. The top-1 query is deemed correct.

Figure 6: Performance of AQQU [6], QUINT [1] and NEQA over the 20 batches of the WQ test set.

very crucial to the overall performance, which explains the low values for the very first batches (the ranking module was trained on a relatively small number of training instances). When user feedback is bypassed, the performance of the baselines deteriorates, as shown in Figure 6b, where the two baselines were not able to improve their performance over time. On the other hand, NEQA showed improvement in performance over time. We attribute this to our *unsupervised* similarity function, which is used to answer questions when the template-based answering mechanism fails and subsequently trigger the learning of new templates with new syntactic structures. The similarity function plays a vital role in distinguishing our system, built with continuous learning in mind, from systems where continuous learning is achieved through simple periodic retraining.

For completeness, we also show results for NEQA when used as a traditional static-learning based QA system, with distinct training and testing phases and with *continuous learning disabled*. Table 3 shows the results for NEQA and baseline systems on the WQ test set after training each on the full WQ training set. The results show that NEQA achieves competitive results, with no significant differences from the best system, but with the added advantage of being able to perform continuous learning when limited training data is available. As described in Sections 2 and 3, NEQA is a continuous-learning extension of Abujabal et al. [1], hence when used in the classical QA setup it obtains the same results as that work.

Method	Avg. Prec.	Avg. Rec.	Avg. F1
Berant et al. [9] (2013)	48.0	41.3	35.7
Yao and Van Durme [60] (2014)	-	-	33.0
Bordes et al. [13] (2014)	-	-	39.2
Bast and Haussmann [6] (2015)	49.8	60.4	49.4
Yih et al. [61] (2015)	52.8	60.7	52.5
Reddy et al. [40] (2016)	-	-	50.3
Savenkov et al. [42] (2016) (w/o text)	49.8	60.4	49.4
Xu et al. [55] (2016) (w/o text)	-	-	47.1
Abujabal et al. [1] (2017)	52.1	60.3	51.0
NEQA	52.1	60.3	51.0

Table 3: Performance of state-of-the-art static learning-based methods on the WebQuestions test set.

Open-domain question answering. NEQA exploits the interaction between syntax and semantics to perform continuous learning on questions coming while the system is deployed. This interaction allows NEQA to perform truly open-domain question answering, where it answers questions that require previously-unseen semantic predicates. To test how well NEQA performs on this task, we restrict the test questions from WQ to 693 questions whose corresponding query contains predicates from the following three domains: `sports`, `government` and `people`. Note that the domain information is encoded in the names of Freebase predicates (e.g., `sports.sports_team.championships`), allowing us to systematically make this restriction. We then removed the 56 questions with queries containing predicates from the above domains from the seed training set used for NEQA (resulting in $300 - 56 = 244$ new seed training examples). To provide a baseline, we used the best publicly available traditional KB-QA system of Bast and Haussmann [6] as a baseline. We train this system on the standard WQ training set with the 1315 questions from the three domains above excluded (a total of $3778 - 1315 = 2463$ training samples).

NEQA achieved an F1 score of 50.3 and 41.5 with and without user feedback, respectively, on the above test set. The system of Bast and Haussmann [6], AQQU, had an F1 score of 20.3. The exhaustive instantiation of the three query templates with all possible KB predicates explains the F1 score achieved by AQQU. This experiment shows that NEQA, in both modes of feedback, answered questions from domains it had never seen during the initial training with a high F1 on par with the results obtained without filtering the seed training set. We attribute these gains to a combination of (i) using templates that account for syntax and using lexicons for instantiating predicate-argument queries as a foundation for NEQA, (ii) re-training the underlying models using test questions which helps NEQA to adapt to the terminology of the new domain, and (ii) the extension of these methods to allow for continuous learning to account for new syntactic structures.

4.3 Analysis

Impact of templates and similarity function. We studied the two branches of NEQA individually: answering with templates, and via the similarity function when the basic answering with templates fails and continuous learning is triggered. Note that we cannot completely decouple both branches since the similarity function feeds our template-based answering with new templates over time, resulting in better coverage and performance. On WQ, with user feedback, 1184 questions were answered with templates, while 848 were answered via the similarity function. For the no-feedback configuration, 1788 out of 2032 were handled by the learned templates, and the similarity function answered 244 questions. The contrast between 1184 and 1788 shows cases where feedback helped weed out erroneous answers obtained through templates.

Components	NEQA	NEQA-No-User-Feedback
Both	**40.8**	**37.0**
Only LM	38.3	35.1
Only word2vec	35.0	33.4

Table 4: F1 scores for a component ablation analysis of our similarity function.

There are various possible failure cases in our pipeline. During the very first batches, the LTR model had a modest ranking performance due to the small number of examples used to train it. However, as more questions were observed, the ranking performance of the LTR model improved substantially, especially when user feedback was harnessed. The impact of this was either incorrect answering, triggering continuous learning, or, once continuous learning was triggered, no answer sets in the top-k being correct, triggering answering via the similarity function. In some cases, none of the generated queries that were fed to the LTR model was correct to start with. This is explained either by the lack of appropriate templates, especially in early batches, or the incompleteness of the underlying lexicons used to instantiate our templates with concrete SPARQL queries (Section 2.3). As future work, we plan to add textual resources to build better lexicons. In other cases, the NERD system failed to link mentions to the correct KB entities.

For some questions, our similarity function failed to retrieve semantically similar questions from our question-query bank. In some of these cases, our bank did not contain any question that is semantically similar to the question at hand. In other cases, although a correct similar question was retrieved, no new template was generated. Again, this is explained by deficiencies in our lexicons, or the NERD system we use.

Similarity function ablation study. Our similarity function consists of two components: (i) a language model (LM), and (ii) *word2vec* similarity. We conducted an ablation study to measure the effect of each component on the overall performance of the system. F1-scores are shown in Table 4. The highest F1 score is achieved when the two components are used. While the LM plays the most vital role, the word2vec component also contributes significantly to the final performance.

Anecdotal results. Table 5 shows sample test questions from the WebQuestions that were correctly answered using templates learned online. These questions represent new syntactic structures that NEQA learned online. Table 6 shows questions that were correctly answered using our similarity function together with the top-1 most similar question retrieved by the similarity function. For

"what is the name of the currency used in italy?"
"what is the head judge of the supreme court called?"
"where did the battle of waterloo occur?"

Table 5: Sample questions correctly answered via templates learned online.

Question:	*"what is the currency in [italy?]"*
Most similar:	*"what kind of money is used in [israel]?"*
Question:	*"what films has [scarlett johansson] been in?"*
Most similar:	*"what movies did [zoe saldana] play in?"*
Question:	*"what was [sir isaac newton]'s inventions?"*
Most similar:	*"what inventions did [robert hooke] made?"*

Table 6: Sample questions correctly answered using the similar questions retrieved from the question-query bank. Entities are generalized using [. . .] placeholders.

every pair of questions in this table, note the differing syntactic structures conveying similar semantics, e.g., *'what is the currency'* and *'what kind of money'*.

5 RELATED WORK

Question answering. KB-QA has seen broad interest in recent years with the wide availability and rapid growth of KBs and voice-based interaction with devices (e.g., Alexa, Cortana). We adopt a template-based approach for mapping syntactic structures to semantic predicate-argument structures [40, 48, 57, 65, 67]. Another way of exploiting syntax is to use grammatical formalisms that derive syntax and semantics in tandem, most prominent among these being CCGs [16, 34]. Some techniques disregard syntax altogether, and rely on combinatorial over-generation of queries followed by a ranking of such candidate queries [6, 9, 60, 61]. Finally, with the recent popularity of deep learning, some methods use large amounts of training data to learn a function for embedding questions and answer entities in a shared latent space [14, 58]. We opted to base NEQA on the systems that use syntax as they achieve better generalization [7], allowing the transfer of semantics between syntactic structures. We rely on dependency parsing for capturing syntax to exploit the rapid progress on this task [17]. In contrast to all the above, our system uses continuous learning that allows starting from small training sets and improving over time.

The framing of the QA task depends on the type of the underlying data and associated annotations. An important QA setting is answering over textual corpora. One way to approach this is using traditional IR methods to retrieve relevant documents and extract passages or phrases that answer the question [15, 26, 41]. Another way has been to use OpenIE to turn such corpora into open-vocabulary knowledge bases and answer over these [23, 24, 33]. Finally, in a setting where both textual and structured data are used, hybrid approaches have been explored for QA [42, 46, 54–56].

In *entity search* [4, 11, 19, 31, 43, 47], the user searches for a list of entities using keyword-based queries (e.g., *'dutch artists paris'*). The underlying corpus for retrieval may be Wikipedia pages [4], documents with Freebase annotations [43], or general RDF-stores [11]. Techniques vary from probabilistic language models [11, 43], query segmentation [31], to category models for entities [4, 47].

Continuous learning and user feedback. Our work draws inspiration from the never-ending learning paradigm [22] and its use case NELL (Never-Ending Language Learning) in machine reading [20]. NEQA also leans on the principle of *online learning* [32] where incoming questions are fed into the system in a sequential order, thus improving the system's performance over time.

User feedback has always been vital for IR systems: be it solely for evaluation as relevance judgments in the early days [49], or in more implicit forms like clicks [30] and reformulations [39] for improving personalized ranking models [2, 3, 8] and automatically completing queries [36]. User interactions play a key role in closing the loop in a continuous learning framework, where they improve the system iteratively. In the NELL system[22], feedback is incorporated as periodic expert judgment on extracted beliefs. Recently, user feedback was leveraged on graph queries, and evaluated with simulated judgments [44]. User feedback has been leveraged in natural language interfaces to databases (NLIDB), where Li et al. [35] invoke user feedback to resolve ambiguous words/phrases in the users' questions, while Iyer et al. [28] ask expert users to provide a full SQL query that answers a question over a database.

Question retrieval. NEQA relies on question retrieval to drive continuous learning when templates fail, where it looks for previously answered questions most similar to the current one. This is a central task in community question answering (CQA) [18, 29, 50, 63, 64, 66], where the goal is to answer a user's question by presenting answers to similar questions that have already been answered. Various methods have been proposed, including those that use syntactic parse trees [50] and language models [63]. Our method takes inspiration from the latter work.

Notions of similarity have been leveraged in KB-QA systems based on paraphrasing. Berant and Liang [10] use a supervised paraphrasing model that finds the logical form whose machine-generated verbalization best paraphrases the input question. Fader et al. [23, 24] perform QA over an open-predicate KB by learning a paraphrase model for rewriting a question to a set of similar canonical question forms, each of which maps to a unique query.

6 CONCLUSION

We presented NEQA, a continuous learning framework for KB-QA that relies on a combination of syntax-aware templates, a semantic similarity function, and judicious invocation of non-expert user feedback on answer sets. NEQA starts with a small seed training set, and exploits failure cases to improve its coverage, by using a similarity function to learn new templates with previously-unseen syntactic structures. Our experiments showed that (i) NEQA has a steady improvement in performance over time, (ii) NEQA is superior to the static learning-based methods with re-training, and (iii) NEQA performs truly open-domain QA.

Improving the similarity function to handle more implicit semantics is a promising direction of future work. Furthermore, we plan to investigate ways to bypass direct user feedback by relying on question reformulations.

REFERENCES

[1] Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated Template Generation for Question Answering over Knowledge Graphs. In *WWW*.

[2] Eugene Agichtein, Eric Brill, and Susan Dumais. 2006. Improving Web search ranking by incorporating user behavior information. In *SIGIR*.

[3] Eugene Agichtein, Eric Brill, Susan Dumais, and Robert Ragno. 2006. Learning user interaction models for predicting Web search result preferences. In *SIGIR*.

[4] Krisztian Balog, Marc Bron, and Maarten de Rijke. 2011. Query modeling for entity search based on terms, categories, and examples. *TOIS* (2011).

[5] Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-Based Question Answering with Knowledge Graph. In *COLING*.

[6] Hannah Bast and Elmar Haussmann. 2015. More Accurate Question Answering on Freebase. In *CIKM*.

[7] Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann A. Copestake. 2015. Layers of Interpretation: On Grammar and Compositionality. In *International Conference on Computational Semantics*.

[8] Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. 2017. Learning from user interactions in personal search via attribute parameterization. In *WSDM*.

[9] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*.

[10] Jonathan Berant and Percy Liang. 2014. Semantic Parsing via Paraphrasing. In *ACL*.

[11] Roi Blanco, Peter Mika, and Sebastiano Vigna. 2011. Effective and Efficient Entity Search in RDF Data. In *ISWC*.

[12] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *SIGMOD*.

[13] Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question Answering with Subgraph Embeddings. In *EMNLP*.

[14] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv* (2015).

[15] Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the AskMSR question-answering system. In *EMNLP*.

[16] Qingqing Cai and Alexander Yates. 2013. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *ACL*.

[17] Danqi Chen and Christopher D. Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *EMNLP*.

[18] Long Chen, Joemon M. Jose, Haitao Yu, Fajie Yuan, and Dell Zhang. 2016. A Semantic Graph based Topic Model for Question Retrieval in Community Question Answering. In *WSDM*.

[19] Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *SIGIR*.

[20] Andrew Carlson et al. 2010. Toward an Architecture for Never-Ending Language Learning. In *AAAI*.

[21] Sören Auer et al. 2007. DBpedia: A Nucleus for a Web of Open Data. In *ISWC*.

[22] Tom M. Mitchell et al. 2015. Never-Ending Learning. In *Conference on AI*.

[23] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-Driven Learning for Open Question Answering. In *ACL*.

[24] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open Question Answering over Curated and Extracted Knowledge Bases. In *KDD*.

[25] Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora. (2013).

[26] Sanda M. Harabagiu, Marius Pasca, and Steven J. Maiorano. 2000. Experiments with Open-Domain Textual Question Answering. In *COLING*.

[27] Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *COLING*.

[28] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. In *ACL*.

[29] Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *CIKM*.

[30] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *KDD*.

[31] Mandar Joshi, Uma Sawant, and Soumen Chakrabarti. 2014. Knowledge Graph and Corpus Driven Segmentation and Answer Inference for Telegraphic Entity-seeking Queries. In *EMNLP*.

[32] Jyrki Kivinen, Alexander J Smola, and Robert C Williamson. 2004. Online learning with kernels. *IEEE Trans. on Signal Processing*.

[33] Jayant Krishnamurthy and Tom M. Mitchell. 2015. Learning a Compositional Semantics for Freebase with an Open Predicate Vocabulary. *TACL* (2015).

[34] Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-Fly Ontology Matching. In *EMNLP*.

[35] Fei Li and Hosagrahar Visvesvaraya Jagadish. 2014. NaLIR: an interactive natural language interface for querying relational databases. In *SIGMOD*.

[36] 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 *WWW*.

[37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv* (2013).

[38] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

[39] Filip Radlinski and Thorsten Joachims. 2005. Query chains: learning to rank from implicit feedback. In *KDD*.

[40] Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *TACL*.

[41] Kiril Ribarov. 2004. Review: Open-Domain Question Answering from Large Text Collections, by Marius Pasca. *Prague Bull. Math. Linguistics* (2004).

[42] Denis Savenkov and Eugene Agichtein. 2016. When a Knowledge Base Is Not Enough: Question Answering over Knowledge Bases with External Text Data. In *SIGIR*.

[43] Uma Sawant and Soumen Chakrabarti. 2013. Learning Joint Query Interpretation and Response Ranking. In *WWW*.

[44] Yu Su, Shengqi Yang, Huan Sun, Mudhakar Srivatsa, Sue Kase, Michelle Vanni, and Xifeng Yan. 2015. Exploiting relevance feedback in knowledge graph search. In *KDD*.

[45] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *WWW*.

[46] Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open Domain Question Answering via Semantic Enrichment. In *WWW*.

[47] Thanh Tran, Philipp Cimiano, Sebastian Rudolph, and Rudi Studer. 2007. Ontology-Based Interpretation of Keywords for Semantic Search. In *ISWC*.

[48] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over RDF data. In *WWW*.

[49] Ellen M. Voorhees. 2001. The Philosophy of Information Retrieval Evaluation. In *CLEF*.

[50] Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009. A syntactic tree matching approach to finding similar questions in community-based QA services. In *SIGIR*.

[51] Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016. Learning Language Games through Interaction. In *ACL*.

[52] Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a Semantic Parser Overnight. In *ACL*.

[53] Keenon Werling, Arun Tejasvi Chaganty, Percy Liang, and Christopher D. Manning. 2015. On-the-Job Learning with Bayesian Decision Theory. In *Conference on Neural Information Processing Systems*.

[54] Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Hybrid Question Answering over Knowledge Base and Free Text. In *COLING*.

[55] Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *ACL*.

[56] Mohamed Yahya, Denilson Barbosa, Klaus Berberich, Qiuyue Wang, and Gerhard Weikum. 2016. Relationship Queries on Extended Knowledge Graphs. In *WSDM*.

[57] Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *CIKM*.

[58] Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint Relational Embeddings for Knowledge-based Question Answering. In *EMNLP*.

[59] Yi Yang and Ming-Wei Chang. 2015. S-MART: Novel Tree-based Structured Learning Algorithms Applied to Tweet Entity Linking. In *ACL*.

[60] Xuchen Yao and Benjamin Van Durme. 2014. Information Extraction over Structured Data: Question Answering with Freebase. In *ACL*.

[61] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *ACL*.

[62] Pengcheng Yin, Nan Duan, Ben Kao, Junwei Bao, and Ming Zhou. 2015. Answering Questions with Complex Semantic Constraints on Open Knowledge Bases. In *CIKM*.

[63] Kai Zhang, Wei Wu, Fang Wang, Ming Zhou, and Zhoujun Li. 2016. Learning Distributed Representations of Data in Community Question Answering for Question Retrieval. In *WSDM*.

[64] Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question Retrieval with High Quality Answers in Community Question Answering. In *CIKM*.

[65] Weiguo Zheng, Lei Zou, Xiang Lian, Jeffrey Xu Yu, Shaoxu Song, and Dongyan Zhao. 2015. How to Build Templates for RDF Question/Answering: An Uncertain Graph Similarity Join Approach. In *Conference on Management of Data*.

[66] Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning Continuous Word Embedding with Metadata for Question Retrieval in Community Question Answering. In *ACL*.

[67] Lei Zou, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over RDF: A graph data driven approach. In *Conference on Management of Data*.