Open Question Answering with Weakly Supervised Embedding Models

Antoine Bordes[†], Jason Weston[‡], and Nicolas Usunier[†]

[†] Université de Technologie de Compiègne – CNRS,
Heudiasyc UMR 7253, Compiègne, France
[‡] Google, 111 8th avenue, New York, NY, USA
`fbordesan,nusunier@utc.fr,jaseweston@gmail.com`

Abstract. Building computers able to answer questions on any subject is a long standing goal of artificial intelligence. Promising progress has recently been achieved by methods that learn to map questions to logical forms or database queries. Such approaches can be effective but at the cost of either large amounts of human-labeled data or by defining lexicons and grammars tailored by practitioners. In this paper, we instead take the radical approach of learning to map questions to vectorial feature representations. By mapping answers into the same space one can query any knowledge base independent of its schema, without requiring any grammar or lexicon. Our method is trained with a new optimization procedure combining stochastic gradient descent followed by a fine-tuning step using the weak supervision provided by blending automatically and collaboratively generated resources. We empirically demonstrate that our model can capture meaningful signals from its noisy supervision leading to major improvements over PARALEX, the only existing method able to be trained on similar weakly labeled data.

Keywords: natural language processing, question answering, weak supervision, embedding models

1 Introduction

This paper addresses the challenging problem of open-domain question answering, which consists of building systems able to answer questions from any domain. Any advance on this difficult topic would bring a huge leap forward in building new ways of accessing knowledge. An important development in this area has been the creation of large-scale Knowledge Bases (KBs), such as FREEBASE [4] and DBPEDIA [15] which store huge amounts of general-purpose information. They are organized as databases of triples connecting pairs of entities by various relationships and of the form (`left entity`, `relationship`, `right entity`). Question answering is then defined as the task of retrieving the correct entity or set of entities from a KB given a query expressed as a question in natural language.

The use of KBs simplifies the problem by separating the issue of collecting and organizing information (i.e. *information extraction*) from the one of searching through it (i.e. *question answering* or *natural language interfacing*). However, open question answering remains challenging because of the scale of these KBs (billions of triples, millions of entities and relationships) and of the difficulty for machines to interpret natural language. Recent progress [6,3,12,10] has been made by tackling this problem with semantic parsers. These methods convert questions into logical forms or database queries (e.g. in SPARQL) which are then subsequently used to query KBs for answers. Even if such systems have shown the ability to handle large-scale KBs, they require practitioners to hand-craft lexicons, grammars, and KB schema for the parsing to be effective. This non-negligible human intervention might not be generic enough to conveniently scale up to new databases with other schema, broader vocabularies or other languages than English.

In this paper, we instead take the approach of converting questions to (uninterpretable) vectorial representations which require no pre-defined grammars or lexicons and can query any KB independent of its schema. Following [10], we focus on answering simple factual questions on a broad range of topics, more specifically, those for which single KB triples stand for both the question and an answer (of which there may be many). For example, (`parrotfish.e`, `live-in.r`, `southern-water.e`) stands for *What is parrotfish's habitat?* and `southern-water.e` and (`cantonese.e`, `be-major-language-in.r`, `hong-kong.e`) for *What is the main language of Hong-Kong?* and `cantonese.e`. In this task, the main difficulties come from lexical variability rather than from complex syntax, having multiple answers per question, and the absence of a supervised training signal.

Our approach is based on learning low-dimensional vector embeddings of words and of KB triples so that representations of questions and corresponding answers end up being similar in the embedding space. Unfortunately, we do not have access to any human labeled (query, answer) supervision for this task. In order to avoid transferring the cost of manual intervention to the one of labeling large amounts of data, we make use of weak supervision. We show empirically that our model is able to take advantage of noisy and indirect supervision by (i) automatically generating questions from KB triples and treating this as training data; and (ii) supplementing this with a data set of questions collaboratively marked as paraphrases but with no associated answers. We end up learning meaningful vectorial representations for questions involving up to 800k words and for triples of an mostly automatically created KB with 2.4M entities and 600k relationships. Our method strongly outperforms previous results on the WikiAnswers+ReVerb evaluation data set introduced by [10]. Even if the embeddings obtained after training are of good quality, the scale of the optimization problem makes it hard to control and to lead to convergence. Thus, we propose a method to fine-tune embedding-based models by carefully optimizing a matrix parameterizing the similarity used in the embedding space, leading to a consistent improvement in performance.

The rest of the paper is organized as follows. Section 2 discusses some previous work and Section 3 introduces the problem of open question answering. Then, Section 4 presents our model and Section 5 our experimental results.

2 Related Work

Large-scale question answering has a long history, mostly initiated via the TREC tracks [22]. The first successful systems transformed the questions into queries which were fed to web search engines, the answer being subsequently extracted from top returned pages or snippets [13,1]. Such approaches require significant engineering to hand-craft queries and then parse and search over results.

The emergence of large-scale KBs, such as Freebase [4] or DBpedia [15], changed the setting by transforming open question answering into a problem of querying a KB using natural language. This is a challenging problem, which would require huge amount of labeled data to be tackled properly by purely supervised machine learning methods because of the great variability of language and of the large scale of KBs. The earliest methods for open question-answering with KBs, based on hand-written templates [25,21], were not robust enough to such variability over possibly evolving KBs (addition/deletion of triples and entities). The solution to gain more expressiveness via machine learning comes from distant or indirect supervision to circumvent the issue of labeled data. Initial works attempting to learn to connect KBs and natural language with less supervision have actually been tackling the information extraction problem [16,11,14,19].

Recently, new systems for learning question answering systems with few labeled data have been introduced based on semantic parsers [6,3,12]. Such works tend to require realistic amounts of manual intervention via labeled examples, but still need vast efforts to carefully design lexicons, grammars and the KB. In contrast, [10] proposed a framework for open question answering requiring little human annotation. Their system, Paralex, answers questions with more limited semantics than those introduced in [3,12], but does so at a very large scale in an open-domain manner. It is trained using automatically and collaboratively generated data and using the KB ReVerb [9]. In this work, we follow this trend by proposing an embedding-based model for question answering that is also trained under weak and cheap supervision.

Embedding-based models are getting more and more popular in natural language processing. Starting from the neural network language model of [2], these methods have now reached near state-of-the-art performance on many standard tasks while usually requiring less hand-crafted features [7,20]. Recently, some embedding models have been proposed to perform a connection between natural language and KBs for word-sense disambiguation [5] and for information extraction [24]. Our work builds on these approaches to instead learn to perform open question answering under weak supervision, which to our knowledge has not been attempted before.

3 Open-domain Question Answering

In this paper, we follow the question answering framework of [10] and use the same data. Hence, relatively little labeling or feature engineering has been used.

3.1 Task Definition

Our work considers the task of question answering as in [10]: given a question q, the corresponding answer is given by a triple t from a KB. This means that we consider questions for which a set of triples t provide an interpretation of the question and its answer, such as:

- q: *What environment does a dodo live in ?*
  t: `(dodo.e, live-in.r, makassar.e)`
- q: *What are the symbols for Hannukah ?*
  t: `(menorah.e, be-for.r, hannukah.e)`
- q: *What is a laser used for?*
  t: `(hologram.e,be-produce-with.r,laser.e)`

Here, we only give a single t per question, but many can exist. In the remainder, the KB is denoted $\mathcal{K}$ and its set of entities and relationships is $\mathcal{E}$. The word vocabulary for questions is termed $\mathcal{V}$. n_v and n_e are the sizes of $\mathcal{V}$ and $\mathcal{E}$ respectively.

Our model consists in learning a function $S(\cdot)$, which can score question-answer triple pairs (q, t). Hence, finding the top-ranked answer $\hat{t}(q)$ to a question q is directly carried out by:

$$\hat{t}(q) = \arg\max_{t' \in \mathcal{K}} S(q, t') \ .$$

To handle multiple answer, we instead present the results as a ranked list, rather than taking the top prediction, and evaluate that instead.

Using the scoring function $S(\cdot)$ allows to directly query the KB without needing to define an intermediate structured logical representation for questions as in semantic parsing systems. We aim at learning $S(\cdot)$, with no human-labeled supervised data in the form (question, answer) pairs, but only by indirect supervision, generated either automatically or collaboratively. We detail in the rest of this section our process for creating training data.

3.2 Training Data

Our training data consists of two sources: an automatically created KB, ReVerb, from which we generate questions and a set of pairs of questions collaboratively labeled as paraphrases from the website WikiAnswers.

Knowledge Base The set of potential answers $\mathcal{K}$ is given by the KB ReVerb [9]. ReVerb is an open-source database composed of more than 14M triples, made of more than 2M entities and 600k relationships, which have been automatically extracted from the ClueWeb09 corpus [17]. In the following, entities are denoted with a `.e` suffix and relationships with a `.r` suffix.

Table 1. Examples of triples from the KB ReVerb.

left entity, relationship, right entity
churchill.e, be-man-of.r, great-accomplishment.e
churchill-and-roosevelt.e, meet-in.r, cairo.e
churchill.e, reply-on.r, may-19.e
crick.e, protest-to.r, churchill.e
churchill.e, leave-room-for.r, moment.e
winston-churchill.e, suffer-from.r, depression.e
churchill.e, be-prime-minister-of.r, great-britain.e
churchill.e, die-in.r, winter-park.e
winston-churchill.e, quote-on.r, mug.e
churchill.e, have-only.r, compliment.e

ReVerb contains broad and general knowledge harvested with very little human intervention, which suits the realistically supervised setting. But, as a result, ReVerb is ambiguous and noisy with many useless triples and entities as well as numerous duplicates. For instance, `winston-churchill.e`, `churchill.e` and even `roosevelt-and-churchill.e` are all distinct entities. Table 3.2 presents some examples of triples: some make sense, some others are completely unclear or useless.

In contrast to highly curated databases such Freebase, ReVerb has more noise but also many more relation types (Freebase has around 20k). So for some types of triple it has much better coverage, despite the larger size of Freebase; for example Freebase does not cover verbs like afraid-of or suffer-from.

Questions Generation We have no available data of questions q labeled with their answers, i.e. with the corresponding triples $t \in \mathcal{K}$. Following [10], we hence decided to create such question-triple pairs automatically. These pairs are generated using the 16 seed questions displayed in Table 2. At each round, we pick a triple at random and then generate randomly one of the seed questions. Note only triples with a `*-in.r` relation (denoted `r-in` in Table 2) can generate from the pattern *where did* `e r` *?*, for example, and similar for other constraints. Otherwise, the pattern is chosen randomly. Except for these exceptions, we used all 16 seed questions for all triples hence generating approximately 16×14M questions stored in a training set we denote $\mathcal{D}$.

The generated questions are imperfect and noisy and create a weak training signal. Firstly, their syntactic structure is rather simplistic, and real questions as posed by humans (such as in our actual test) can look quite different to them. Secondly, many generated questions do not correspond to semantically valid English sentences. For instance, since the type of entities in ReVerb is unknown, a pattern like *who does* `e r` *?* can be chosen for a triple where the type of ? in `(?, r, e)` is not a person, and similar for other types (e.g. *when*). Besides, for the strings representing entities and relationships in the questions, we simply used their names in ReVerb, replacing `-` by spaces and stripping off

Table 2. Patterns for generating questions from ReVerb triples following [10].

KB Triple	Question Pattern	KB Triple	Question Pattern
`(?, r, e)`	*who* `r e` *?*	`(?, r, e)`	*what is e's* `r` *?*
`(?, r, e)`	*what* `r e` *?*	`(e, r, ?)`	*who is* `r` *by* `e` *?*
`(e, r, ?)`	*who does* `e r` *?*	`(e, r-in, ?)`	*when did* `e r` *?*
`(e, r, ?)`	*what does* `e r` *?*	`(e, r-on, ?)`	*when did* `e r` *?*
`(?, r, e)`	*what is the* `r` *of* `e` *?*	`(e, r-in, ?)`	*when was* `e r` *?*
`(?, r, e)`	*who is the* `r` *of* `e` *?*	`(e, r-on, ?)`	*when was* `e r` *?*
`(e, r, ?)`	*what is* `r` *by* `e` *?*	`(e, r-in, ?)`	*where was* `e r` *?*
`(?, r, e)`	*who is e's* `r` *?*	`(e, r-in, ?)`	*where did* `e r` *?*

their suffixes, i.e. the string representing `winston-churchill.e` is simply *winston churchill*. While this is often fine, this is also very limited and caused many incoherences in the data. Generating questions with a richer KB than ReVerb, such as Freebase or DBpedia, would lead to better quality because typing and better lexicons could be used. However, this would contradict one of our motivations which is to train a system with as little human intervention as possible (and hence choosing ReVerb over hand-curated KBs).

Paraphrases The automatically generated examples are useful to connect KB triples and natural language. However, they do not allow for a satisfactory modeling of English language because of their poor wording. To overcome this issue, we again follow [10] and supplement our training data with an indirect supervision signal made of pairs of question paraphrases collected from the WikiAnswers website.

On WikiAnswers, users can tag pairs of questions as rephrasing of each other. [10] harvested a set of 18M of these question-paraphrase pairs, with 2.4M distinct questions in the corpus. These pairs have been labeled collaboratively. This is cheap but also causes the data to be noisy. Hence, [10] estimated that only 55% of the pairs were actual paraphrases. The set of paraphrases is denoted $\mathcal{P}$ in the following. By considering all words and tokens appearing in $\mathcal{P}$ and $\mathcal{D}$, we end up with a size for the vocabulary $\mathcal{V}$ of more than 800k.

4 Embedding-based Model

Our model ends up learning vector embeddings of symbols, either for entities or relationships from ReVerb, or for each word of the vocabulary.

4.1 Question-KB Triple Scoring

Architecture Our framework concerns the learning of a function $S(q, t)$, based on embeddings, that is designed to score the similarity of a question q and a triple t from $\mathcal{K}$.

Our scoring approach is inspired by previous work for labeling images with words [23], which we adapted, replacing images and labels by questions and triples. Intuitively, it consists of projecting questions, treated as a bag of words (and possibly n-grams as well), on the one hand, and triples on the other hand, into a shared embedding space and then computing a similarity measure (the dot product in this paper) between both projections. The scoring function is then:

$$S(q,t) = \boldsymbol{f}(q)^\top \boldsymbol{g}(t)$$

with $\boldsymbol{f}(\cdot)$ a function mapping words from questions into $\mathbb{R}^k$, $\boldsymbol{f}(q) = \boldsymbol{V}^\top \varPhi(q)$. $\boldsymbol{V}$ is the matrix of $\mathbb{R}^{n_v \times k}$ containing all word embeddings $\boldsymbol{v}$, $\varPhi(q)$ is the (sparse) binary representation of q ($\in \{0,1\}^{n_v}$) indicating absence or presence of words. Similarly, $\boldsymbol{g}(\cdot)$ is a function mapping entities and relationships from KB triples into $\mathbb{R}^k$, $\boldsymbol{g}(t) = \boldsymbol{W}^\top \varPsi(t)$, $\boldsymbol{W}$ the matrix of $\mathbb{R}^{n_e \times k}$ containing all entities and relationships embeddings $\boldsymbol{w}$, and $\varPsi(q)$ the (sparse) binary representation of t ($\in \{0,1\}^{n_e}$) indicating absence or presence of entities and relationships.

Representing questions as a bag of words might seem too limited, however, in our particular setup, syntax is generally simple, and hence quite uninformative. A question is typically formed by an interrogative pronoun, a reference to a relationship and another one to an entity. Besides, since lexicons of relationships and entities are rather disjoint, even a bag of words representation should lead to decent performance, up to lexical variability. There are counter-examples such as *What are cats afraid of ?* vs. *What are afraid of cats ?* which require different answers, but such cases are rather rare. Future work could consider adding parse tree features or semantic role labels as input to the embedding model.

Contrary to previous work modeling KBs with embeddings (e.g. [24]), in our model, an entity does not have the same embedding when appearing in the left-hand or in the right-hand side of a triple. Since, $\boldsymbol{g}(\cdot)$ sums embeddings of all constituents of a triple, we need to use 2 embeddings per entity to encode for the fact that relationships in the KB are not symmetric and so that appearing as a left-hand or right-hand entity is different.

This approach can be easily applied at test time to score any (question, triple) pairs. Given a question q, one can predict the corresponding answer (a triple) $\hat{t}(q)$ with:

$$\hat{t}(q) = \arg\max_{t' \in \mathcal{K}} S(q,t') = \arg\max_{t' \in \mathcal{K}} \big(\boldsymbol{f}(q)^\top \boldsymbol{g}(t')\big).$$

Training by Ranking Previous work [23,24] has shown that this kind of model can be conveniently trained using a ranking loss. Hence, given our data set $\mathcal{D} = \{(q_i, t_i), i = 1, \dots, |\mathcal{D}|\}$ consisting of (question, answer triple) training pairs, one could learn the embeddings using constraints of the form:

$$\forall i,\ \forall t' \neq t_i,\quad \boldsymbol{f}(q_i)^\top \boldsymbol{g}(t_i) > 0.1 + \boldsymbol{f}(q_i)^\top \boldsymbol{g}(t')\ \ ,$$

where 0.1 is the margin. That is, we want the triple that labels a given question to be scored higher than other triples in $\mathcal{K}$ by a margin of 0.1. We also enforce a constraint on the norms of the columns of $\boldsymbol{V}$ and $\boldsymbol{W}$, i.e. $\forall_i, ||\boldsymbol{v}_i||_2 \leq 1$ and $\forall_j, ||\boldsymbol{w}_j||_2 \leq 1$.

To train our model, we need positive and negative examples of (q, t) pairs. However, $\mathcal{D}$ only contains positive samples, for which the triple actually corresponds to the question. Hence, during training, we use a procedure to corrupt triples. Given $(q, t) \in \mathcal{D}$, we create a corrupted triple t' with the following method: pick another random triple t_{tmp} from $\mathcal{K}$, and then, replace with 66% chance each member of t (left entity, relationship and right entity) by the corresponding element in t_{tmp}. This heuristic creates negative triples t' somewhat similar to their positive counterpart t, and is similar to schemes of previous work (e.g. in [7,5]).

Training the embedding model is carried out by stochastic gradient descent (SGD), updating $\boldsymbol{W}$ and $\boldsymbol{V}$ at each step. At the start of training the parameters of $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ (the $n_v \times k$ word embeddings in $\boldsymbol{V}$ and the $n_e \times k$ entities and rel. embeddings in $\boldsymbol{W}$) are initialized to random weights (mean 0, standard deviation $\frac{1}{k}$). Then, we iterate the following steps to train them:

1. Sample a positive training pair (q_i, t_i) from $\mathcal{D}$.
2. Create a corrupted triple t'_i ensuring that $t'_i \neq t_i$.
3. Make a stochastic gradient step to minimize $\big[0.1 - \boldsymbol{f}(q_i)^\top \boldsymbol{g}(t_i) + \boldsymbol{f}(q_i)^\top \boldsymbol{g}(t'_i)\big]_+$.
4. Enforce the constraint that each embedding vector is normalized.

The learning rate of SGD is updated during the course of learning using ADAGRAD [8]. $\big[x\big]_+$ is the positive part of x.

Multitask Training with Paraphrases Pairs We multitask the training of our model by training on pairs of paraphrases of questions (q_1, q_2) from $\mathcal{P}$ as well as training on the pseudolabeled data constructed in $\mathcal{D}$. We use the same architecture simply replacing $\boldsymbol{g}(\cdot)$ by a copy of $\boldsymbol{f}(\cdot)$. This leads to the following function that scores the similarity between two questions:

$$S_{\text{prp}}(q_1, q_2) = \boldsymbol{f}(q_1)^\top \boldsymbol{f}(q_2) \ .$$

The matrix $\boldsymbol{W}$ containing embeddings of words is shared between S and S_{prp}, allowing it to encode information from examples from both $\mathcal{D}$ and $\mathcal{P}$. Training of S_{prp} is also conducted with SGD (and ADAGRAD) as for S, but, in this case, negative examples are created by replacing one of the questions from the pair by another question chosen at random in $\mathcal{P}$.

During our experiments, $\boldsymbol{W}$ and $\boldsymbol{V}$ were learned by alternating training steps using S and S_{prp}, switching from one to another at each step. The initial learning rate was set to 0.1 and the dimension k of the embedding space to 64. Training ran for 1 day on a 16 core machine using HOGWILD [18].

4.2 Fine-tuning the Similarity between Embeddings

The scale of the problem forced us to keep our architecture simple: with $n_e \approx$ 3.5M (with 2 embeddings for each entity) and $n_v \approx$ 800k, we have to learn around 4.3M embeddings. With an embedding space of dimension $k = 64$, this leads to around 275M parameters to learn. The training algorithm must also stay simple to scale on a training set of around 250M of examples ($\mathcal{D}$ and $\mathcal{P}$ combined); SGD appears as the only viable option.

SGD, combined with ADAGRAD for adapting the learning rate on the course of training, is a powerful algorithm. However, the scale of the optimization problem makes it very hard to control and conduct properly until convergence. When SGD stops after a pre-defined number of epochs, we are almost certain that the problem is not fully solved and that some room for improvement remains: we observed that embeddings were able to often rank correct answers near the top of the candidates list, but not always in the first place.

In this paper, we introduce a way to fine-tune our embedding-based model so that correct answers might end up more often at the top of the list. Updating the embeddings involves working on too many parameters, but ultimately, these embeddings are meant to be used in a dot-product that computes the similarity between q and t. We propose to learn a matrix $\boldsymbol{M} \in \mathbb{R}^{k \times k}$ parameterizing the similarity between words and triples embeddings. The scoring function becomes:

$$S_{\mathrm{ft}}(q,t) = \boldsymbol{f}(q)^\top \boldsymbol{M} \boldsymbol{g}(t) \ .$$

$\boldsymbol{M}$ has only k^2 parameters and can be efficiently determined by solving the following convex problem (fixing the embedding matrices $\boldsymbol{W}$ and $\boldsymbol{V}$):

$$\min_{\boldsymbol{M}} \frac{\lambda}{2} \parallel \boldsymbol{M} \parallel_F^2 + \frac{1}{m} \sum_{i=1}^{m} \left[1 - S_{\mathrm{ft}}(q_i, t_i) + S_{\mathrm{ft}}(q_i, t_i')\right]_+^2 \ ,$$

where $\parallel \boldsymbol{X} \parallel_F$ is the Frobenius norm of $\boldsymbol{X}$. We solve this problem in a few minutes using L-BFGS on a subset of $m = 10$M examples from $\mathcal{D}$. We first use 4M examples to train and 6M as validation set to determine the value of the regularization parameter λ. We then retrain the model on the whole 10M examples using the selected value, which happened to be $\lambda = 1.7 \times 10^{-5}$.

This fine-tuning is related to learning a new metric in the embedding space, but since the resulting $\boldsymbol{M}$ is not symmetric, it does not define a dot-product. Still, $\boldsymbol{M}$ is close to a constant factor times identity (as in the original score $S(\cdot)$). The fine-tuning does not deeply alter the ranking, but, as expected, allows for a slight change in the triples ranking, which ends in consistent improvement in performance, as we show in the experiments.

5 Experiments

5.1 Evaluation Protocols

We first detail the data and metrics which were chosen to assess the quality of our embedding model.

Test Set The data set WIKIANSWERS+REVERB contains no labeled examples but some are needed for evaluating models. We used the test set which has been

Table 3. Performance of variants of our embedding models and Paralex [10] for reranking question-answer pairs from the WikiAnswers+ReVerb test set.

Method	**F1**	**Prec**	**Recall**	**MAP**
Paralex *(No. 2-arg)*	0.40	**0.86**	0.26	0.12
Paralex	0.54	0. 77	0.42	0.22
Embeddings	0.68	0.68	0.68	0.37
Embeddings *(no paraphrase)*	0.60	0.60	0.60	0.34
Embeddings *(incl. n-grams)*	0.68	0.68	0.68	0.39
Embeddings+fine-tuning	**0.73**	0.73	**0.73**	**0.42**

created by [10] in the following way: (1) they identified 37 questions from a held-out portion of WikiAnswers which were likely to have at least one answer in ReVerb, (2) they added all valid paraphrases of these questions to obtain a set of 691 questions, (3) they ran various versions of their paralex system on them to gather candidate triples (for a total of 48k), which they finally hand-labeled.

Reranking We first evaluated different versions of our model against the paralex system in a reranking setting. For each question q from the WikiAnswers+ReVerb test set, we take the provided candidate triples t and rerank them by sorting by the score $S(q,t)$ or $S_{\text{ft}}(q,t)$ of our model, depending whether we use fine-tuning or not. As in [10], we then compute the precision, recall and F1-score of the highest ranked answer as well as the mean average precision (MAP) of the whole output, which measures the average precision over all levels of recall.

Full Ranking The reranking setting might be detrimental for paralex because our system simply never has to perform a full search for the good answer among the whole ReVerb KB. Hence, we also conducted an experiment where, for each of the 691 questions of the WikiAnswers+ReVerb test set, we ranked all 14M triples from ReVerb. We labeled the top-ranked answers ourselves and computed precision, recall and F1-score.[1]

Table 4. Performance of our embedding model for retrieving answers for questions from the WikiAnswers+ReVerb test set, among the whole ReVerb KB (14M candidates).

Method	**F1**
Embeddings	0.16
Embeddings+fine-tuning	0.22
Embeddings +string-matching	0.48
Embeddings+fine-tuning+string-matching	**0.57**

[1] We provide the top-ranked answers and our labels as supplementary material.

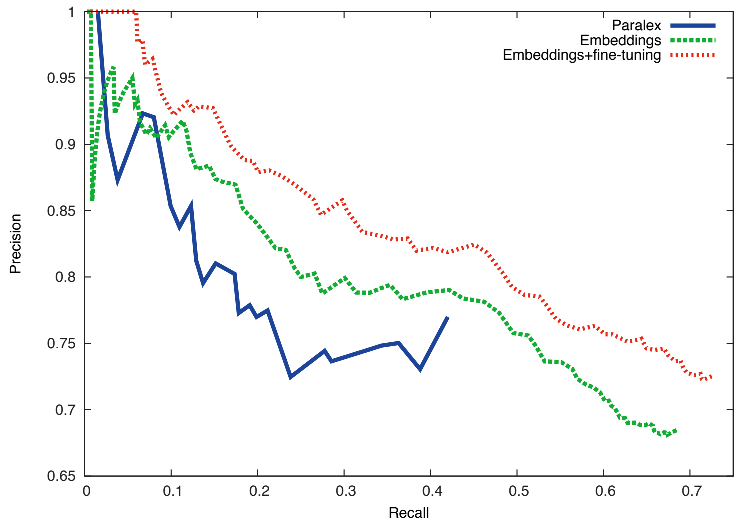

Fig. 1. Precision-recall curves of our embedding model and Paralex [10] for reranking question-answer pairs from the WikiAnswers+ReVerb test set.

5.2 Results

This section now discusses our empirical performance.

Reranking Table 3 and Figure 1 present the results of the reranking experiments. We compare various versions of our model against two versions of paralex, whose results were given in [10].

First, we can see that multitasking with paraphrase data is essential since it improves F1 from 0.60 to 0.68. Paraphrases allow for the embeddings to encode a richer connection between KB constituents and words, as well as between words themselves. Note that the WikiAnswers data provides word alignment between paraphrases, which we did not use, unlike paralex. We also tried to use n-grams (2.5M most frequent) as well as the words to represent the question, but this did not bring any improvement, which might at first seem counter-intuitive. We believe this is due to two factors: (1) it is hard to learn good embeddings for n-grams since their frequency is usually very low and (2) our automatically generated questions have a poor syntax and hence, many n-grams in this data set do not make sense. We actually conducted experiments with several variants of our model, which tried to take the word ordering into account (e.g. with convolutions), and they all failed to outperform our best performance without word order, once again perhaps because the supervision is not clean enough to allow for such elaborated language modeling. Fine-tuning the embedding model is very beneficial to optimize the top of the list and grants a bump of 5 points of F1: carefully tuning the similarity makes a clear difference.

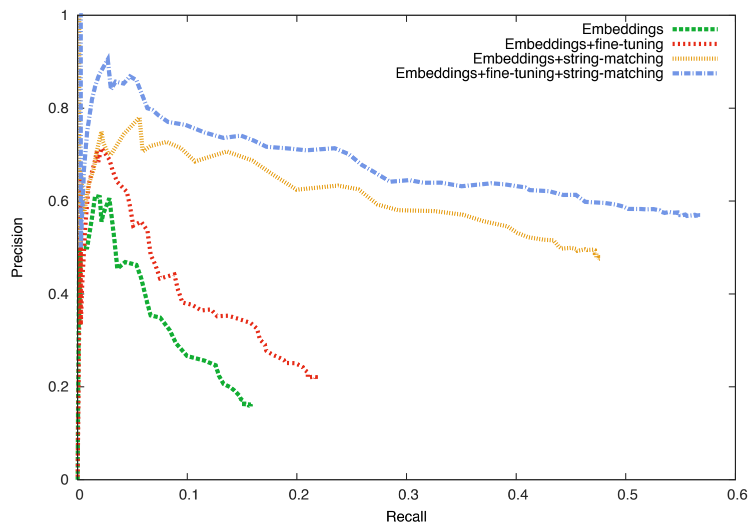

Fig. 2. Precision-recall curves for retrieving answers for questions from the WikiAnswers+ReVerb test set, among the whole ReVerb KB (14M candidates).

All versions of our system greatly outperform paralex: the fine-tuned model improves the F1-score by almost 20 points and, according to Figure 1, is better in precision for all levels of recall. paralex works by starting with an initial lexicon mapping from the KB to language and then gradually increasing its coverage by iterating on the WikiAnswers+ReVerb data. Most of its predictions come from automatically acquired templates and rules: this allows for a good precision but it is not flexible enough across language variations to grant a satisfying recall. Most of our improvement comes from a much better recall.

However, as we said earlier, this reranking setting is detrimental for paralex because paralex was evaluated on the task of reranking some of its own predictions. The results provided for paralex, while not corresponding to those of a full ranking among all triples from ReVerb (it is still reranking among a subset of candidates), concerns an evaluation setting more complicated than for our model. Hence, we also display the results of a full ranking by our system in the following.

Full Ranking Table 4 and Figure 2 display the results of our model to rank all 14M triples from ReVerb. The performance of the plain models is not good (F1 $= 0.22$ only for S_{ft}) because the ranking is degraded by too many candidates. But most of these can be discarded beforehand.

We hence decided to filter out some candidates before ranking by using a simple string matching strategy: after pos-tagging the question, we construct a set of candidate strings containing (i) all noun phrases that appear less than 1,000

Table 5. Examples of nearest neighboring entities and relationships from REVERB for some words from our vocabulary. The prefix `L:`, resp. `R:`, indicates the embedding of an entity when appearing in **`left-hand`** side, resp. **`right-hand`** side, of triples.

Word	**Closest entities or relationships from ReVerb in the embedding space**
get rid of	`get-rid-of.r be-get-rid-of.r rid-of.r can-get-rid-of.r will-get-rid-of.r should-get-rid-of.r have-to-get-rid-of.r want-to-get-rid-of.r will-not-get-rid-of.r help-get-rid-of.r`
useful	`be-useful-for.r be-useful-in.r R:wide-range-of-application.e can-be-useful-for.r be-use-extensively-for.r be-not-very-useful-for.r R:plex-or-technical-algorithm.e R:internal-and-external-use.e R:authoring.e R:good-or-bad-purpose.e`
radiation	`R:radiation.e L:radiation.e R:gamma-radiation.e L:gamma-radiation.e L:x-ray.e L:gamma-ray.e L:cesium-137.e R:electromagnetic-radiation.e L:external-beam-radiation.e L:visible-light.e`
barack-obama	`L:president-elect-barack-obama.e L:barack-obama.e R:barack-obama.e L:president-barack-obama.e L:obama-family.e L:sen.-barack-obama.eL:president-elect-obama.e R:president-barack-obama.e L:democratic-presidential-candidate-barack-obama.e L:today-barack-obama.e`
iphone	`R:iphone.e L:iphone.e R:t-mobile.e R:apple-iphone.e L:lot-of-software.e L:hotmail.e R:windows-mobile-phone.e L:skype.e R:smartphone.e R:hd-dvd-player.e`

times in ReVerb, (ii) all proper nouns if any, otherwise the least frequent noun phrase in ReVerb. This set of strings is then augmented with the singular form of plural nouns, removing the final "s", if any. Then, only the triples containing at least one of the candidate strings are scored by the model. On average, about 10k triples (instead of 14M) are finally ranked for each question, making our approach much more tractable.

As expected, string matching greatly improves results, both in precision and recall, and also significantly reduces evaluation time.

The final F1 obtained by our fine-tuned model is even better then the result of PARALEX in reranking, which is pretty remarkable, because this time, this setting advantages it quite a lot.

Embeddings Table 5 displays some examples of nearest neighboring entities from ReVerb for some words from our vocabulary. As expected, we can see that verbs or adverbs tend to correspond to relationships while nouns refer to entities. Interestingly, the model learns some synonymy and hyper/hyponymy. For instance, *radiation* is close to `x-ray.e` and *iphone* to `smartphone.e`. This happens thanks to the multitasking with paraphrase data, since in our automatically generated (q, t) pairs, the words *radiation* and *iphone* are only used for entities with the strings `radiation` and `iphone` respectively in their names.

Table 6. Performance of our embedding model for retrieving answers for questions from the WebQuestions test set, among the whole ReVerb KB (14M candidates).

Method	**Top-1**	**Top-10**	**F1**
Emb.	0.025	0.094	0.025
Emb.+fine-tuning	0.032	0.106	0.032
Emb. +string-match.	0.085	0.246	0.068
Emb.+fine-tuning+string-match.	**0.094**	**0.270**	**0.076**

5.3 Evaluation on WebQuestions

Our initial objective was to be able to perform open-domain question answering. In this last experimental section, we tend to evaluate how generic our learned system is. To this end, we propose to ask our model to answer questions coming from another dataset from the literature, but without retraining it with labeled data, just by directly using the parameters learned on WikiAnswers+ReVerb.

We chose the data set WebQuestions [3], which consists of natural language questions matched with answers corresponding to entities of Freebase: in this case, no triple has to be returned, only a single entity. We used exact string matching to find the ReVerb entities corresponding to the Freebase answers from the test set of WebQuestions and obtained 1,538 questions labeled with ReVerb out of the original 2,034.

Results of different versions of our model are displayed in Table 6. For each test question, we record the rank of the first ReVerb triple containing the answer entity. Top-1 and Top-10 are computed on questions for which the system returned at least one answer (around 1,000 questions using string matching), while F1 is computed for all questions. Of course, performance is not great and can not be directly compared with that of the best system reported in [3] (more than 0.30 of F1). One of the main reasons is that most questions of WebQuestions, such as *Who was vice-president after Kennedy died?*, should be represented by multiple triples, a setting for which our system has not been designed. Still, for a system trained with almost no manual annotation nor prior information on another dataset, with an other –very noisy– KB, the results can be seen as particularly promising. Besides, evaluation is broad since, in ReVerb, most entities actually appear many times under different names as explained in Section 3. Hence, there might be higher ranked answers but they are missed by our evaluation script.

6 Conclusion

This paper introduces a new framework for learning to perform open question answering with very little supervision. Using embeddings as its core, our approach can be successfully trained on imperfect labeled data and indirect supervision and significantly outperforms previous work for answering simple factual questions. Besides, we introduce a new way to fine-tune embedding models for cases where their optimization problem can not be completely solved.

In spite of these promising results, some exciting challenges remain, especially in order to scale up this model to questions with more complex semantics. Due to the very low supervision signal, our work can only answer satisfactorily simple factual questions, and does not even take into account the word ordering when modeling them. Further, much more work has to be carried out to encode the semantics of more complex questions into the embedding space.

References

1. M. Banko, E. Brill, S. Dumais, and J. Lin. Askmsr: Question answering using the worldwide web. In *Proceedings of 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, 2002.
2. Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.
3. J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, October 2013.
4. K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008.
5. A. Bordes, X. Glorot, J. Weston, and Y. Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *Proc. of the 15th Intern. Conf. on Artif. Intel. and Stat.*, volume 22. JMLR W&CP, 2012.
6. Q. Cai and A. Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, August 2013.
7. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.
8. J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12, 2011.
9. A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing (EMNLP '11)*, Edinburgh, Scotland, UK, July 27-31 2011.
10. A. Fader, L. Zettlemoyer, and O. Etzioni. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1608–1618, Sofia, Bulgaria, 2013.
11. R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, 2011.
12. T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, October 2013.
13. C. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)*, 19(3), 2001.
14. N. Lao, A. Subramanya, F. Pereira, and W. W. Cohen. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.
15. J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 2014.

16. M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proc. of the Conference of the 47th Annual Meeting of ACL*, 2009.
17. J. Pomikálek, M. Jakubícek, and P. Rychlỳ. Building a 70 billion word corpus of english from clueweb. In *LREC*, pages 502–506, 2012.
18. B. Recht, C. Ré, S. J. Wright, and F. Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS 24).*, 2011.
19. S. Riedel, L. Yao, A. McCallum, and B. M. Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*, 2013.
20. R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.
21. C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over rdf data. In *Proceedings of the 21st international conference on World Wide Web*, 2012.
22. E. M. Voorhees and D. M. Tice. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2000.
23. J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1), 2010.
24. J. Weston, A. Bordes, O. Yakhnenko, and N. Usunier. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371, 2013.
25. M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. Natural language questions for the web of data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2012.