SN Computer Science (2021) 2:25
https://doi.org/10.1007/s42979-020-00421-7

ORIGINAL RESEARCH

Question Answering Over Knowledge Base: A Scheme for Integrating Subject and the Identified Relation to Answer Simple Questions

Happy Buzaaba[1] · Toshiyuki Amagasa[2]

Received: 3 March 2020 / Accepted: 8 December 2020 / Published online: 9 January 2021

Abstract
Answering natural language question over a knowledge base is an important and challenging task with a wide range of application in natural language processing and information retrieval. Several existing knowledge-based question answering systems exploit complex end-to-end neural network approaches that are computationally expensive and take long to execute when training the neural network. More importantly, such an end-to-end approach makes it difficult to examine the process of query processing. In this study, we decompose the question answering problem in a three-step pipeline of entity detection, entity linking, and relation prediction, and solve each component separately. We explore basic neural network and non-neural network methods for entity detection and relation prediction plus a few heuristics for entity linking. We also introduce a method to identify ambiguity in the data and show that ambiguity in the data bounds the performance of the question answering system. The experiment on the SimpleQuestions benchmark data set shows that a combination of basic LSTMs, GRUs, and non-neural network techniques achieve reasonable performance while providing an opportunity to understand the question answering problem structure.

Keywords Question answering · Knowledge base · Neural networks · Simple questions

Introduction

Question answering overs knowledge base has been conducted using large-scale knowledge bases (KB), such as Freebase [6], DBpedia [26], Wikidata [34], and YAGO [18]. These knowledge bases consist of a large pool of information with real-world entities as nodes and relations as edges. Each directed edge, along with its head entity and tail entity, constitute a triple, i.e., head entity, relation, and tail entity, which is also named as a fact. Knowledge bases have in recent years become pivotal in question answering because of their ability to provide precise answers to users' questions. Users commonly use structured query languages like SPARQL for querying resource description framework (RDF) data in such knowledge bases. SPARQL is a powerful query language which requires expert knowledge that is hard to learn for non-programmers who would want to access the information in the KB. Therefore, the most convenient approach that has gained much attention is knowledge-based question answering that allows end users to pose natural language questions over a knowledge base without knowledge of the underlying schema, and in return receive entities as the answers to the question.

In the past, a number of question answering systems, such as AquaLog [28], NLP-Reduce [22], PowerAqua [29], and FREyA [10], have been proposed. Many of them map a natural language question to a triple-based representation. For example, a simple question like "who wrote the Neverending story?", PowerArqua [29] would map this question to the triple representation *([person,organisation], wrote, Neverending story)*. Then, similarity measures are applied to retrieve the matching sub-graphs from the RDF repository. This approach, however, has a number of drawbacks such as failure to capture the original semantic structure of the question using triples.

In addition, several recent studies in deep learning have seen a surge of end-to-end complex neural network approaches that have performed well on a variety of natural language processing tasks like opinion extraction [21],

✉ Happy Buzaaba
happy-b@kde.cs.tsukuba.ac.jp

1 Systems and Information Engineering University of Tsukuba, Tsukuba, Japan

2 Center for Computational Sciences, University of Tsukuba, Tsukuba, Japan

sentence classification [23], and entity linking [19, 35] to mention but a few. Such end-to-end architectures suffer from long execution time when training the networks and more so difficult to conduct detailed performance analysis in the end-to-end setting.

Based on the existing SimpleQuestions benchmark [7], where only a single fact from the knowledge base is required to answer a question, our work reduces the question answering task to finding a single fact, i.e., subject, predicate, and object, in the KB that answers the question. For example, given a question "where was Barack Obama born?", the aim of our work is to identify the corresponding triple, i.e., *(Barack Obama, people/person/place/of/birth, Honolulu)* from the knowledge base. Simple questions are the most common types of questions observed in various question answering sites [35]. Although this task is refereed to as "simple", in reality, it is difficult and far from being solved. We, therefore, propose a method that identifies a fact that matches a question's best from the knowledge base. The proposed method bear a resemblance to answer selection [36] in which the question answering system chooses the answer from a candidate list given a question.

To address this issue, we formulate the question answering task as a fact selection problem where the system matches a candidate fact to the question. The task is decomposed into three different components: (1) entity detection, (2) relation prediction, and (3) entity linking. Freebase [6] is used as the KB in our experiment where entities are represented with machine identifiers (MIDs). A key challenge in knowledge base question answering is to find the matching entities and relations in the given knowledge base. The entity linking problem is hard, because the question may use a synonymy or variant of the name used in the KB, and the knowledge base may contain many entity machine identifiers (MIDs) with the same name. For example, answering a question 'what country was the film the debt from?' becomes difficult, because there are 4 entity MIDs (04j0t75, 0bj3wz4j, 0bjwlk1l, and 0dy60p) with the name 'the debt' in the Freebase KB. Relation matching faces a similar problem of multiple relations (film/film/country, film/film/written_by, music/album/release_type) and its not clear which one is being referenced. These ambiguity problems exacerbate for large-scale knowledge bases. It becomes crucial to avoid restrictions on lexical matches, since we might miss out on the correct query, and if we allow weaker matches, the number of possibilities might rise. We introduce a method to identify such ambiguities in the data.

In summary, this paper makes the following contributions:

- We present a simple yet effective approach to address the knowledge base question answering task. Our approach is faster, efficient, and performs reasonably well compared to previous complex approaches of Bodes et al. [7], Golub and He [15], and Lukovinikov et al. [30], which apply end-to-end neural network on a similar task of simple question answering.
- We establish a strong non-neural-network baseline on this task to compare with neural networks. the baseline includes CRF's (Conditional Random Fields) for entity detection and LR (Logistic Regression) for relation classification.
- We show that ambiguity in the data limits the performance. There are often multiple answers that are not easy to disambiguate. We introduce a method to identify such ambiguities in the data which improves the performance from 74.64 to 79.9% a 5.26% increase.
- Finally, we present an empirical error analysis to gain insights into the mistakes produced and the reasons that bring about the mistakes in an attempt to improve the performance in the future work.

The rest of the paper is organised as follows: We formulate the problem and introduce key concepts in the section "Problem statement and definitions"; we summarize related works in the section "Related work", before providing the details of our approach in the section "Proposed approach". We examine experiments and provide a detailed analysis and discussion in the section "Experiments", and conclude the article in the section "Conclusion and future work" with an outlook on future work.

Problem Statement and Definitions

The goal of our study is to design a question answering system that can map a simple natural language question q to a matching query Q consisting of the subject and predicate/relation referred to in the question that can be executed against the knowledge base G to retrieve the answer to the question. For the purpose of experiment, we restrict ourselves to simple questions, which only require the retrieval of a single fact from the knowledge base to be answered.

Knowledge Base A knowledge base G comprises of triples of the form (s, p, o) where s, p, and o denote the head entity, predicate/relation, and the tail entity, respectively. A triple can also be interpreted as a directed edge, from s the head entity to o the tail entity labeled with a relation p. The Freebase knowledge base (KB) is used to provide facts/triples for answering the questions in the simple question data set.

SimpleQuestions The SIMPLEQUESTIONS data set [7] is a common benchmark used for studying the simple question answering task. The task is to generate a query of the form subject–relation pair denoted as (s, p) which can be executed against the KB to retrieve the answer to the question. The query subject can be traced as Freebase object with a machine identifier (MID) which typically include one or more string aliases for example (mid 04j0t75 is named “the debt”). The query relation is a Freebase object property defined by the ontology (i.e., film/film/country). The query for the question “what country was the film the debt from?” corresponds to the subject–relation pair (04j0t75 [the debt], film/film/country). The SimpleQuestions data set consists of 108,442 simple questions, annotated with a ground truth Freebase triple (s, p, o) each. We use the FB2M Freebase subset in our experiments for purposes of comparison with previous work that applied a similar subset on the simple question answering task.

We formally define the knowledge base question answering task as: given a knowledge base $G = \{(s_i, p_i, o_i)\}$ that represents a set of triples, and a natural language question $q = \{w_1, w_2, \dots, w_T\}$, where $w_i \in q$ is a sequence of words, the simple question answering task is to find a triple $(\hat{s}, \hat{p}, \hat{o}) \in G$, such that $\hat{o}$ is the answer to the question.

Related Work

Question answering over knowledge base has gained much attention in recent years, and increasingly, researchers are building end-to-end neural network models for this task. In this section, we will review some of the existing work on the task.

Several existing methods directly parse the natural language question into a structured query using semantic parsing, and these include syntactic parsing [5], semantic role labeling [4], semantic parsing [14], as well as mapping a question to a logical form and then translating it to a structured query [2, 3]. These approaches, however, are language domain-specific, since they rely on linguistic heuristics and hand-crafted features which prove to be a down side, since they are not able to generalize well.

Our work focuses on answering simple factoid questions which require the extraction of a single fact from a structured knowledge base (KB) to be answered.

In [7], the *SimpleQuestions* benchmark was first introduced by Bodes et al., and this benchmark consists of 108,442 simple questions annotated with the correct Freebase knowledge base triple or fact. All facts have exactly one possible relation. In his work, Bodes et al. proposed a memory network as the original solution to solve the task. This benchmark prompted a line of work that have led several researchers like Golub and He [15], Lukovinikov et al. [30], and Dai et al. [9] to apply even more complex neural network architectures to address this problem.

In their work, Golub and He [15] apply a character-level attention-based encoder–decoder model to the simple question answering task. This approach was first developed by [1] for machine translation and it introduces attention for handling longer sequences which proved to be suitable for tasks like semantic parsing [11]. However, the architecture by Golub and He is purely based on character level which results in longer input sequences. Compared to machine translation where we expect longer sequences, attention mechanism would be of lesser importance in simple question answering where only two predictions need to be made for every input sequence. Our model is more efficient and avoids the attention mechanism complexity. We believe, however, that addition of attention mechanism may be advantageous in future when we extend our model to more complex questions.

In [35], authors use character-level convolutional neural network for entity linking and a separate word-level convolutional neural network with attentive max-pooling that models the relationship between the predicate and question pattern to improve the model. This approach employs attention mechanism to obtain better matches for the relations. Our work only focuses on word-level encoding and does not apply attention mechanisms.

Furthermore, Dai et al. [9] investigated a word-level recurrent neural network (RNN) where they proposed a conditional probabilistic framework using bidirectional gated recurrent units (BiGRUs) to infer the target relation first and then the target subject associated with the candidate relations. This work is trained on Freebase-specific predicate and entity representation, and can less easily be transferred to other KB’s.

In addition, Lukovinikov et al. [30] apply a hierarchical word-level and character-level question encoder to train a neural network. This model learns to rank subject–predicate pairs in an end-to-end manner to enable the retrieval of relevant facts given a question. The above techniques exploit increasingly complex end-to-end deep learning techniques that are computationally expensive and limit the opportunity to fully understand the problem structure. Other methods that have tried to address quite similar problems include [24, 33, 37], although these methods require expensive manual annotation.

Our goal is to enhance the performance of the simple question answering system using baseline methods. We take a different approach of decomposing the question answering task into different components and solve each component separate. This approach performs reasonably well compared to complex neural network methods and provides the opportunity to understand the problem structure. We conduct a study to examine the contribution of complex models for the simple question answering task by exploring the performance of baseline methods.

Proposed Approach

In this work, we propose a three-step pipeline, as illustrated in Fig. 1, to address the simple question answering task:

- *Step 1 (Entity detection)* The objective is to tag each individual word in the question as either entity or non-entity. We apply standard Recurrent Neural Networks (RNNs), and Conditional Random Fields (CRFs) to identify entities in the question.
- *Step 2 (Relation prediction)* We classify the question as one of the relation types in the KB by applying standard (RNNs), Convolutional Neural Networks (CNNs), plus logistic regression.
- *Step 3 (Entity linking).* We then link the identified entities to there corresponding nodes in the KB by constructing an inverted index using python dictionaries to generate a candidate list of triples/fact with their respective score and finally rely on the relation from the classifier to filter out the candidate list and retain the best candidate triple with the object entity as the answer.

Our initial goal is to generate a structured query Q (subject, relation) from the natural language question q that accurately interprets the question. To generate the structured query, our approach makes two assumptions; first, we assume first-order questions that can be answered by retrieving a single fact from the KB, and second, we assume that the source entity/subject entity is mentioned in the question.

Step 1: Entity Detection

The goal of Step 1 is to tag each individual word in the question as either entity or non-entity. In the Simplequestions data set, each question is associated with a triple (subject, predicate, and object) from a Freebase subset that answers the question. The subject is given as an MID; we use the names file [9] which consists of entity MID's and their corresponding entity names to check entity mentions in the question and annotate tokens as either entity or non-entity. In some cases, there were no exact matches which introduced some noise. We apply fuzzy matching to project the entity mention to n-gram sequence with the smallest edit distance to the entity name. Edit distance [27] is a common string matching metric for measuring the difference between two sequences. Informally, edit distance between two words is a minimum number of single-character edits (insertion, deletions, or substitutions) required to change one word into another. The edit distance between two strings d, e of length $|d|$ and $|e|$, respectively, is given by $D_{d,e}(|d|, |e|)$:

$$D_{d,e}(i,j) = \min\begin{cases} D_{d,e}(i-1,j)+1 \\ D_{d,e}(i,j-1)+1 \\ D_{d,e}(i-1,j-1)+1_{(d_i \neq e_j)}, \end{cases} \quad \text{otherwise.} \tag{1}$$

where $1_{(d_i \neq e_j)}$ is the indicator function equivalent to 0 when $d_i = e_j$ and equal to 1 otherwise, and $D_{d,e}$ is the distance between the first i characters of d and the first j characters of e. i and j are 1-based indices. The first element in the minimum corresponds to deletion from d to e, the second to insertion, and third to match or mismatch depending on whether the respective symbols are the same.

We identify the Entity being queried by formulating the entity detection task as a sequence labeling problem, as shown in Fig. 2, where each word or token is tagged as entity or non-entity; I: entity and 0: non-entity. To this end, we apply both recurrent neural network and conditional random field.

Recurrent Neural Network (RNN) Recurrent neural networks [12] are capable of processing arbitrary sequential inputs by applying an activation function to the hidden vector recursively. In Fig. 2, we represent each question word/token with a word embedding, and the representation is then combined with the hidden layer representation from the previous time step using either BiLSTMs (Bidirection Long Short-Term Memory) [17] or BiGRUs (Bidirectional Gated Recurrent Units) [8].

The key idea of LSTMs is the cell state. As we read the sentence from left to right, the LSTM is going to have a new memory variable $\mathbf{C}^{\langle t \rangle}$ called the memory cell at time step $\mathbf{t}$,

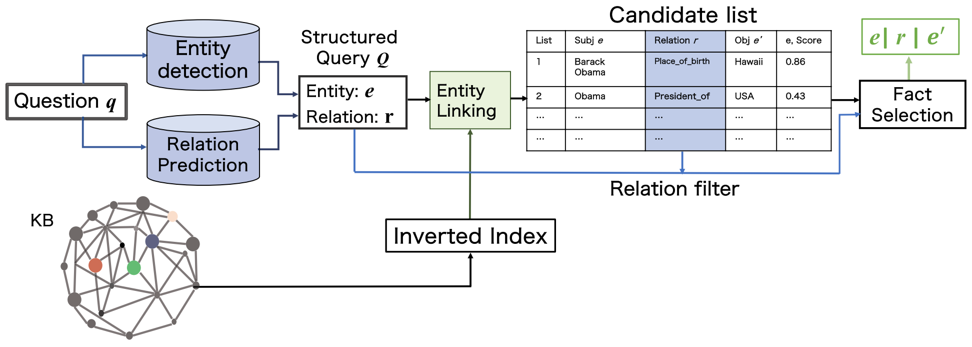

Fig. 1 Illustration of the proposed approach, the two main components of entity detection and relation prediction compose Q the structured query from which a list of candidates tripples is generated by an inverted index from the knowledge base G, and the list is filtered using the query relation type to select the best candidate

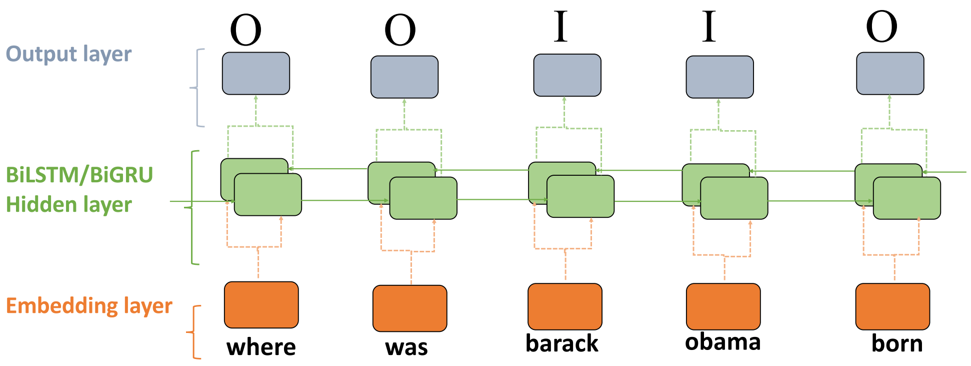

Fig. 2 Sequence tagging for entity detection

so that when the network gets further into the sentence, it can still remember information seen earlier [16]. LSTMs have the ability to control the hidden state updates and outputs using gates. They consist of three gating functions; the update gate Γ_{u} used to control the update at each time step, the forget gate Γ_{f} which decides the amount of information from the previous hidden state to keep or throw away, and the output gate Γ_{o} which regulates the flow of information in and out of the cell.

The current memory cell $\mathbf{C}^{\langle t\rangle}$ is computed by interpolating the previous hidden state $\mathbf{a}^{\langle t-1\rangle}$ and the candidate state $\tilde{\mathbf{C}}^{\langle t\rangle}$. The equations that govern the LSTM behavior are defined as:

$$\begin{aligned}
\hat{\mathbf{C}}^{\langle t\rangle} &= tanh(W_c[\mathbf{a}^{\langle t-1\rangle}, X^{\langle t\rangle}] + b_c)\\
\Gamma_{\mathrm{u}} &= \sigma(W_{\mathrm{u}}[\mathbf{a}^{\langle t-1\rangle}, X^{\langle t\rangle}] + b_{\mathrm{u}})\\
\Gamma_{\mathrm{f}} &= \sigma(W_{\mathrm{f}}[\mathbf{a}^{\langle t-1\rangle}, X^{\langle t\rangle}] + b_{\mathrm{f}})\\
\Gamma_{\mathrm{o}} &= \sigma(W_{\mathrm{o}}[\mathbf{a}^{\langle t-1\rangle}, X^{\langle t\rangle}] + b_{\mathrm{o}})\\
\mathbf{C}^{\langle t\rangle} &= \Gamma_{\mathrm{u}} \odot \hat{\mathbf{C}}^{\langle t\rangle} + \Gamma_{\mathrm{f}} \odot \mathbf{C}^{\langle t-1\rangle}\\
\mathbf{a}^{\langle t\rangle} &= \Gamma_{\mathrm{o}} \odot \mathbf{C}^{\langle t\rangle},
\end{aligned} \tag{2}$$

where $\hat{\mathbf{C}}^{\langle t\rangle}$ is the candidate cell state at time $\mathbf{t}$, $\mathbf{a}^{\langle t-1\rangle}$ the activation at previous time step, $X^{\langle t\rangle}$ the current input, and W and b are parameter and bias terms, respectively. $\mathbf{C}^{\langle t\rangle}$ is the current internal cell state with $\odot$ as the element-wise vector product. σ denotes sigmoid and tanh denotes the hyperbolic tangent.

The GRUs, on the other hand, consists of two gates; the update gate Γ_{u}, and the reset gate Γ_{r}. The update gate allows the model to learn by itself, how much of the previous information should be passed to the future. This limits the vanishing gradient problem, since the model does not have to remember all the information seen previously. The reset gate tells how relevant is the previous cell state for computing the current candidate state. The GRU transition equations are defined as follows:

$$\begin{aligned}
\hat{\mathbf{C}}^{\langle t\rangle} &= tanh(W_c[\Gamma_{\mathrm{r}} \odot \mathbf{C}^{\langle t-1\rangle}, X^{\langle t\rangle}] + b_c)\\
\Gamma_{\mathrm{u}} &= \sigma(W_{\mathrm{u}}[\mathbf{C}^{\langle t-1\rangle}, X^{\langle t\rangle}] + b_{\mathrm{u}})\\
\Gamma_{\mathrm{r}} &= \sigma(W_{\mathrm{r}}[\mathbf{C}^{\langle t-1\rangle}, X^{\langle t\rangle}] + b_{\mathrm{r}})\\
\mathbf{C}^{\langle t\rangle} &= \Gamma_{\mathrm{u}} \odot \hat{\mathbf{C}}^{\langle t\rangle} + (1-\Gamma_{\mathrm{u}}) \odot \mathbf{C}^{\langle t-1\rangle},
\end{aligned} \tag{3}$$

whereby $\mathbf{C}^{\langle t-1\rangle}$ is the previous cell state and σ denotes the sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$ that is applied element wise to the vector.

The BiLSTMs and BiGRUs apply a non-linear transformation to compute the hidden layer representation at the current time step. The final hidden representation at the current time step is then projected to the output dimensional space and normalized into a probability distribution via a softmax layer, as shown in Fig. 2.

Conditional Random Fields (CRFs) The other method that we use for entity detection is Conditional random fields [25] so as to have a performance comparison between non-neural networks with neural networks on the entity detection task. CRFs are used to represent the probability of a hidden state sequence given some observations. For example, given $x = (x_1, x_2, \dots, x_m)$ as the input sequence, and $s = (s_1, s_2, \dots, s_m)$ the output states or crf tags, the conditional probability c_p can be given by $c_p = p(s|x)$. We define $\Phi(x,s) \in R^d$ a feature map that maps x paired with s to d a dimensional feature vector. The probability is therefore modeled as a log linear:

$$p(s|x,w) = \frac{\exp(w \cdot \phi(x,s))}{\sum_{s'} \exp(w \cdot \phi(x,s'))}, \tag{4}$$

where $w \in R^d$ is a parameter vector with s' ranging over all possible outputs. The parameter vector w can be estimated by assuming that we have a set of n labeled samples $\{(x^i, s^i)\}^n$, with $i = 1$. The regularized log likelihood is given by:

$$\mathcal{L}(w) = \sum_{i=1}^{n} \log p(s^i|x^i, w) - \frac{\lambda 2}{2}||w||_2^2 - \lambda 1||w||_1, \quad (5)$$

where $\frac{\lambda 2}{2}||w||_2^2$ and $\lambda_1||w||_1$ forces w to be small in the respective norm. We estimate a parameter vector $w*$ as $w^* = \text{argmax}(w \in R^d \mathcal{L}(w))$. If we are able to estimate $w*$ the parameter vector, we can then find the most likely tag a sentence $s*$ for a sentence x by $s^* = \text{argmax}_s p(s|x : w^*)$.

We use Stanford Named Entity Recognizer (Stanford NER) [13] to train crf. This tool labels word sequences in the sentence into four classes; person, organization, location, and non-entity. It extracts features such as current/previous and next word, part of speech (POS) tag, character n-gram etc, and trains a crf model. In this work, the question is tagged into four classes, the first three classes of (person, organization, and location) were tagged as entity. In the experiment, we trained the Stanford NER on the training set and labeled the test set questions. After classifying every token as entity (denoted by **I**) or non-entity (denoted by **0**), entity phrases are extracted by grouping consecutive words, i.e., given a question *"where was barack obama born?"*, the entity detection output is [**0 0 I I 0**], from which a single entity "barack obama" is extracted.

Step 2: Relation Prediction

In step 1, we obtain the subject entity from the question words. The goal of step 2 is to identify the relation being queried in the given natural language question to come up with the structured query. In this step, the entire natural language question is classified as one of the knowledge base relation types. The Freebase knowledge base consists of 1837 unique relation types (possible labels). We therefore conduct a large-scale classification to assign the relation type to the question. To achieve this, we explore several methods including RNNs, convolutional neural networks (CNNs), and logistic regression. We introduce logistic regression, so that we can compare the performance of neural networks and non-neural network on the relation prediction task.

Recurrent Neural network (RNN) A model similar to the one used for entity detection is used, and both BiLSTM and BiGRU are applied. However, relation prediction is not a tagging task, since it is over the entire question. The classification decision is based on the output of the hidden layer of the last token, as shown in Fig. 3.

Convolutional Neural Networks (CNNs) We also apply vanilla CNNs to extract local features by sliding filters over the word embeddings.

In Fig. 4, the sentence is represented by concatenating words and padding where necessary as $x_{1:n} = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, and we use convolutional filters on the input matrix transformed into word embeddings to generate new features from a window of words represented by $c_i = f(W \cdot x_{i:i+h-1} + b)$. The filter is applied to each of the possible window of words in the sentence to produce a feature map represented as $c = [c_1, c_2, \ldots, c_{n-h+1}]$ and we apply maxPooling over the filter to take the maximum value as a feature corresponding to this particular filter. The idea is to capture the most important feature, which is basically one with the highest value for each feature map. And finally, these features are passed on to the fully connected softmax layer whose output is the probability distribution over labels. CNNs have shown to perform well on sentence classification [20]. In our experiment, we use the architecture in [23], by changing it to a single static channel instead, and use it for relation classification.

Logistic Regression To compare the performance of neural network with non-neural network methods on the relation prediction task, we apply logistic regression. In our experiment, we consider two types of features extracted from the question:

(1) *Term Frequency-Inverse Document Frequency (tfidf)* To extract *tfidf* question features, we apply existing tools *CountVectorizer* and *TfidfTransformer* sci-kit learn classes.
(2) *Question Word embeddings and 1-Hot encoding of relation words* To obtain the question representation, both question and relation words are used. Table 1 shows sample questions and the respective relations. Question word embeddings are averaged and out-of-vocabulary words are assigned a vector of zero. Words

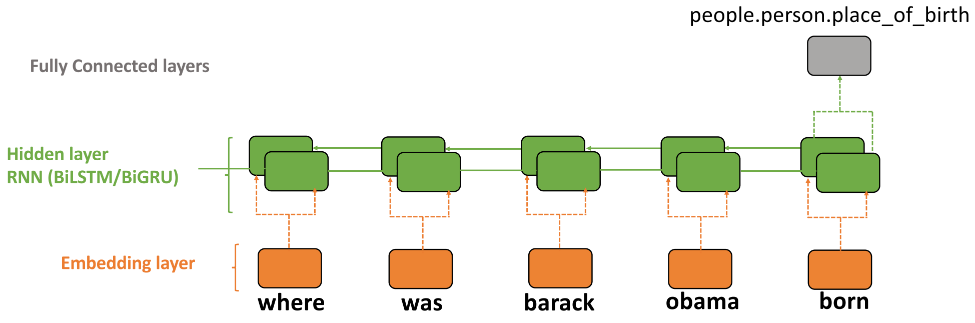

Fig. 3 RNN architecture for relation prediction

in the relation class are split into individual tokens to come up with a vocabulary of relation words.

In Fig. 5, the vector representations are concatenated to come up with the question features. This vector representation combines word embeddings strength and one-hot encoding. Both the neural network and the non-neural network methods classify the entire question into one of the Freebase KB relation types.

Finally, a structured query consisting of the entity from step 1, and the relation from step 2, is generated. Using a similar example of the question "where was Barack Obama born?", the relation prediction from step 2 is [*people/person/place_of_birth*]. If the identified entity from step 1 is *"Barack Obama"*,

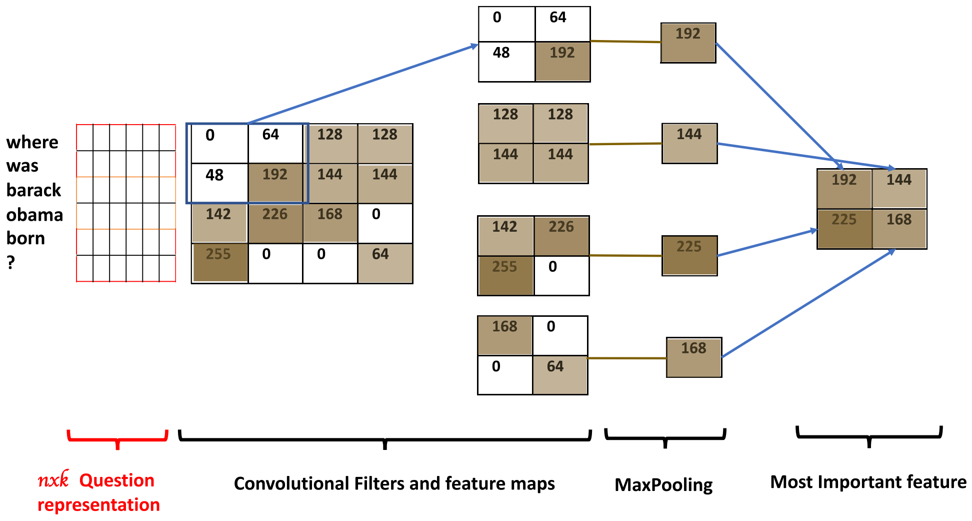

Fig. 4 How convolutional filters are applied to word window to produce a feature map to which Maxpooling is applied to reduce the size while maintaining the most important features for Relation prediction

Table 1 Sample questions and their relation, the questions, and relations are split into individual words to come up with their respective representations that is to say word embeddings from question words and bag of words from relation words

Question	Relation
What is the place of birth of Barack Obama born?	People/person/place of birth
What country was the film the debt from?	Film/film/country
Which artist recorded most of us are sad?	Music/recording/artist

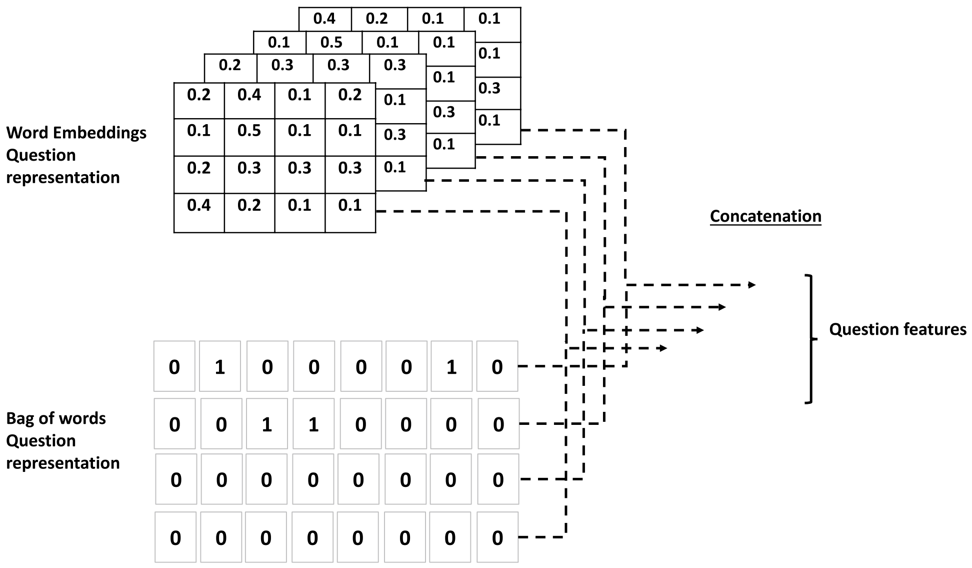
Word Embeddings Question representation
Bag of words Question representation
Concatenation
Question features

Fig. 5 Concatenation of the average word embeddings from the question words and the bag of words from the relation words to come up with the question features

the structured query $\{entity, relation\}$ becomes $\{\}\}BarackObama", people/person/place\_of\_birth\}$.

Step 3: Entity Linking

From Step 1 and Step 2, we generate the structured query that accurately represents the natural language question. In step 3, the main objective is to identify the correct entity node in the KB. The detected entity words in step 1 that represent the candidate entity in the structured query are linked to the actual entity node in the knowledge base. We build indexes to link the extracted entity to the actual knowledge base: first, a names index which maps all entity (MID's) in the Freebase subset to their names in the names file [9]; second, the inverted index to map any entity n-gram to all nodes in the knowledge base.

Figure 6 shows the entity linking process; we iterate over entity word n-grams in a decreasing order of n; if we find candidate values, early termination is applied to stop searching for smaller values of n. If, for example, our query *'Barack Obama'* for $n = 2$ finds candidate entities, then the search will be terminated and only those entities will be included. This helps in pruning entities that would have been retrieved for queries $'barack'$ and $'obama'$ for $n = 1$. After coming up with all possible entities in Freebase, candidate entity nodes are retrieved from the index and appended to the list. These candidate entities in the list are then scored using *inverse document frequency (idf)* and ranked in descending order.

Once we have a list of candidate entities, each candidate node is used as a starting point to reach candidate answers. We limit our search to a single hop and retrieve all nodes that are reachable from the candidate node where the relation path is consistent with the predicted relation (in the structured query). We look at the relation types, and all those candidate entity nodes with relation type different from the one generated in step 2 are removed. Only those candidate entity nodes with a relation type leading to another node that is similar to the one generated in step 2 are kept from which the entity node with a high score in the remaining candidate list has an object entity node which is the answer to the question.

Identification of Ambiguity in the Data

One of the contributions of this work is to show that there exist multiple answers to the question that are not easy to disambiguate which limits the performance of the question answering system. This is a common challenge in free open data sets and such ambiguities are most likely to arise from the annotation process. In the Simplequestions data set, annotators are asked to write a natural language question for a corresponding triple [7]. In such a case, where one is only given a triple, it would be difficult to anticipate possible ambiguities in the KB. We identify such ambiguities in the data.

Given a natural language question q with the corresponding triple (s, p, o), where $s, p, ando$ are the subject, relation, and object, respectively, we aim at determining a set of all possible (s, p) pairs that accurately interpret the question q. The first step is to determine the string alias a by matching the phrase in q (i.e., entity in the structured query) with the subject alias s in freebase. Next, we find all other Freebase entity MID's that share this alias and add them to a set S.

For example, given the question *'what country was the film the debt from?'*, in Table 2, we can see three examples of subject–relation pairs with equal linguistic evidence that cannot be easily disambiguated. After generating a set of entities S, the next step is to come up with a set of potential relations P, as shown in Table 2. For this, we abstract a from the question, i.e., *"what country was the film $\langle e \rangle$ from?"* to determine an abstract relation p. An accurate semantic interpretation of the question q is defined if there exists a

Table 2 Examples of ambiguity in the data

Entity MID (S)	Freebase alias (s)	Potential relations (P)
fb:m.04j0t75	The debt (country)	film/film/country
fb:m.0bj3wz4j	The debt (film)	film/film/written_by
fb:m.0bjwlk11	The debt (music)	music/album/release_type

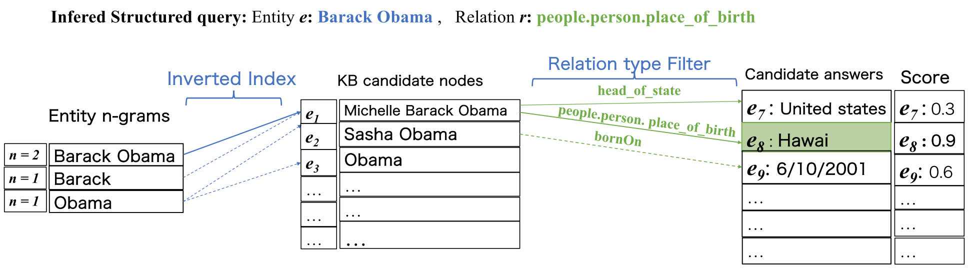

Fig. 6 Illustration of the entity linking process. The inverted index map entity n-grams to corresponding nodes in the knowledge base to come up with a candidate list. The relation type filter is used to filter out all candidate nodes with a different relation from that in the structured query

subject–relation pair $(s, p) \in KB$ where $p \in P$ and $s \in S$. In a case where multiple (s, p) pairs exist like in (Table 2), the question is not answerable. To answer such a question, we predict the most likely relation $p_{\max} \in P$ that makes the answer candidate to be $(s, p_{\max}) \in$ KB. If it happens that answer candidates are more, i.e., p_{max} is more than one, then we pick $s_{\max}$ with the most facts of type $p_{\max}$.

Experiments

Set-Up

We experiment on the *SimpleQuestions* [7] benchmark and we use the *2M Freebase subset* [6] as the knowledge base. The *Simplequestions* data set consists of 108,442 examples, with 75,910 , 10,845, and 21,687, training, validation, and testing samples, respectively. Each question is associated with a triple from freebase that answers the question.

In our experiment, word embeddings are initialized using glove [31] vectors of 300-dimension, and we use negative log likelihood to optimize parameters using Adam with the initial learning rate of 0.0001 with batch size of 64. We compute precision, recall, and F1 for every sequence tags against the ground truth for evaluation in entity detection, and we evaluate recall for top results ($R@k$) for both entity linking and relation prediction to see if the correct answer appears in the top k results and the final prediction is marked as correct if both entity and relation match the ground truth in end-to-end evaluation.

We implement each component (entity detection and relation prediction) of our model in PyTorch v0.2.1, and use the fuzzywuzzy package to compute string matching scores on a single CPU 3.3 GHz core i5 macOS sierra.

Results and Discussion

In this section, we discuss our results for the simple question answering task on each individual component.

Entity Detection We evaluate the precision, recall, and F1-score on the token span level. This means that the predicted entity token span exactly matches the ground truth (a true positive span). The results in Table 3 reveal that BiGRU and BiLSTM perform better with F1-score of 92.63% for the BiLSTM. It can also be noticed that the non-neural network crf result of 90.2% is comparable with the neural network.

Entity Linking The results in Table 4 shows the performance of a given model on the entity linking task. We observe that, at $R@1$, the performance of BiLSTM and BiGRU was not different. It is, however, very interesting to see that the result by CRF a non-neural network method is comparable to the neural network with a 0.013 difference. Although CRF may have performed slightly lower than the BiLSTM and BiGRU on entity detection, the bottleneck is entity linking.

On trying to understand why the entity linking performance was far below compared to the entity detection result, we observed a number of ambiguous entities in the knowledge base that have got a similar label hence making it difficult to identify the correct entity MID.

Table 3 Entity detection results for both neural network and non-neural network methods

Metrics	Neural net				Non-neural net	
	BiGRU		BiLSTM		CRF	
	Val	Test	Val	Test	Val	Test
Precision (%)	92.69	92.12	92.73	92.04	90.85	90.72
Recall (%)	93.28	93.12	93.46	93.23	89.92	89.8
F1 (%)	92.99	92.62	93.09	92.63.	90.36	90.2

Table 4 Entity linking results for both neural network and non-neural network methods

R@k	Neural net				Non-neural net	
	BiGRU		BiLSTM		CRF	
	Val	Test	Val	Test	Val	Test
1	0.675	0.662	0.679	0.662	0.663	0.649
3	0.789	0.776	0.793	0.776	0.776	0.762
5	0.823	0.81	0.827	0.811	0.809	0.796
10	0.86	0.849	0.889	0.876	0.871	0.861
20	0.885	0.876	0.912	0.903	0.895	0.889
50	0.909	0.903	0.912	0.903	0.895	0.889

Table 5 Relation prediction results for both neural network and non-neural network methods

R @ k	Neural net					
	BiGRU		BiLSTM		CNN	
	Val	Test	Val	Test	Val	Test
1	82.22	81.59	81.61	81.10	82.88	81.92
3	93.75	93.68	93.59	93.35	93.75	93.68
5	95.93	95.76	96.10	95.52	95.86	95.64
R @ k	Non-neural net					
	LR(TF-IDF)		LR(W2Vec+1-Hot)			
	Val	Test	Val	Test		
1	73.36	73.64	71.64	72.31		
3	85.67	86.39	86.7	87.11		
5	88.58	89.16	90.12	91.63		

Relation Prediction Table 5 shows the performance of different models on the relation prediction task. We observe that, at $R@1$, CNN outperforms both BiGRU and BiLSTM. We, however, can see that neural network methods (i.e., BiGRU, BiLSTM, and CNN) perform much better than non-neural network (i.e., Logistic Regression) on the relation prediction task. It is also observed that logistic regression performs better on test set than the validation set, and this is mainly due to the fact that we used logistic regression with default parameters.

Another important observation is that training each of the models component, i.e., entity detection and relation prediction for 50 epochs using our PC, and it takes approximately 8 h which is quicker and efficient compared to training Lukovinikov et al. [30] which takes close to 6 days for the same number of epochs.

Fact Selection Table 6 shows test set accuracy results for various combinations of entity detection and relation prediction. Our best model combination that achieves 74.64% is BiLSTM for entity detection and BiGRU for relation prediction. This result improves to [[79.9%]] when data ambiguity is removed. When we replace Neural net with CRFs for entity detection, the accuracy decreases by 1.25%. Our results show a reasonable performance on non-neural network methods. A combination of CRFs for entity detection with logistic regression (W2Vec+1-Hot or

Table 6 Accuracy results for selecting the correct fact from the knowledge base that has the object entity which answers the question

Entity detection	Relation prediction	Accuracy (%)	R@2	R@3
BiLSTM	**BiGRU**	74.64 [[79.9]]	80.93	82.75
BiLSTM	CNN's	74.63	80.85	82.75
CRF	CNN	73.42	79.45.	81.3
CRF's	**BiGRU**	73.39	79.46	81.28
CRF's	BiLSTM	73.34	79.36	81.16
CRF's	**LR(W2Vec+1-Hot)**	70.1	77.07	79.08
CRF's	LR(TF-IDF)	68.4	...	...
Previous work				
Model	Description	Accuracy		
Yin et al. 2016	Max-pooling	76.4		
Dai et al. 2016	Probabilistic	75.7		
Xiao et al. 2019	Embedding based	75.4		
Lukovinikov 2017	Neural embedding	71.2		
Golub and He 2016	Character-based	70.9		
Bodes et al. 2015	Memory network	62.7		

In the table, we compare our result with previous state of the art methods that apply complex end-to-end methods

TFIDF) for relation prediction achieves 70.1% and 68.4%, respectively.

One could argue that LR(W2Vec+1-Hot) still takes advantage of W2Vec a neural net, but LR(TF-IDF) is entirely a non-neural network baseline and performs way higher than the original paper [7] which applied complex memory networks. On comparing our results with other existing state-of-art complex models to examine the necessity of model complexity on this task, we observe that our best results outperform the complex neural network models of Golub and He's [15] and Lukovinikov [30]. However, our model, is comparable to Dai et al. [9] and Yin et al. [35] which apply a separately trained segmentation. Our best accuracy is less than two points away from the next highest reported result in the literature.

Considering the above comparisons, our results suggest that despite the immense contribution of neural networks in improving the state of the art on the simple questions data set, the improvement directly attributed to complex neural networks is modest. We also believe that it is important to pay attention when interpreting the results due to non-determinism associated with training neural networks that can yield differences in the accuracy [32].

Error Analysis

In an attempt to improve the performance, we manually inspected, so that we can understand some of the errors that arise in our model, and their main cause. We considered the questions that were retrieved in the second position (hits = 2) but not in the top position, and we found that 733 questions were retrieved in the second position. We also considered question that were retrieved in the third position (hits = 3) but not in the first and second positions, and we found 203 questions. We performed the analysis using RNN for entity detection and RNN for relation prediction using the top 50 entities and the top 5 relations for answering the question on the validation set. We found out that errors mainly occurred during entity linking and Fig. 7 shows the causes of the error.

In general, the two main reasons for the error were incorrect query which was due to errors propagated during the entity detection phase and incorrect linking where it was noticed that in almost all cases, different MIDs had the same entity name and the model was unable to disambiguate the correct entity from the incorrect one. We observed 33.9% of such examples (3675 of 10,846) in the simplequestions data set. Such examples that were found to have multiple MIDs with the same name include the name "holywood", with 270 entity MIDs, the subject name 'chalie chaplin', with over 20 entity MIDs, etc. Such cases made it difficult to answer the questions. To overcome this, the relation frequency for each subject in the KB was taken into account (see: "Identification of ambiguity in the data") which improves the performance of our model to a new accuracy of 79.9%.

Another observation was a number of questions that did not reference a subject. For example, a question *"which book is written about?"* does not refer the corresponding subject 01n7q:California. It was also observed that many questions where a correct MID was seen but with a wrong relation were hard to disambiguate even for humans. For example, the question *"which release was reading on?"* refers to a relation $\}music/release\_track/recording'$ the classifier, however, predicted $\}music/release\_track/release'$.

Conclusion and Future Work

In this work, we present evidence that ambiguities in the data limit performance on the simple-questions data set and we present a method to deal with it. Our main contribution in this work is establishing strong baselines one that uses simple neural net and one without neural-net. We are convinced that our two baselines will encourage future researchers to adequately examine and take advantage of simple baselines to fully understand the simple-questions problem structure. Our approach applies simple baseline methods, but achieves results comparable to state-of-the-art methods that apply complex neural networks and attention mechanisms. This is surprising given the proven strength of attention mechanisms. Although we did not apply attention mechanisms on

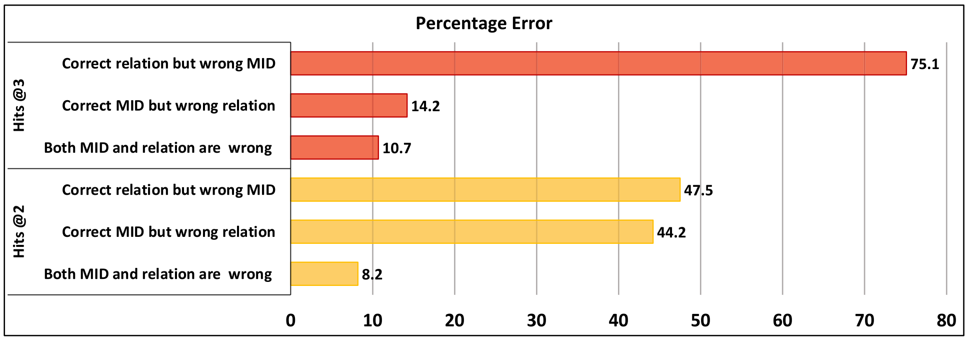

Fig. 7 Main causes of error that limit the performance of our model

the simple questions task, we believe that they might be of help in future work when we extend our approach to more complex questions.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. 2014. arXiv preprint arXiv:1409.0473.
2. Berant J, Chou A, Frostig R, Liang P. Semantic parsing on freebase from question-answer pairs. In: Proceedings of the 2013 conference on empirical methods in natural language processing; 2013, pp. 1533–44.
3. Berant J, Liang P. Semantic parsing via paraphrasing. In: Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: long papers); 2014. pp. 1415–25
4. Bilotti MW, Elsas J, Carbonell J, Nyberg E. Rank learning for factoid question answering with linguistic and semantic constraints. In: Proceedings of the 19th ACM international conference on Information and knowledge management. ACM; 2010. pp. 459–68.
5. Bilotti MW, Ogilvie P, Callan J, Nyberg E. Structured retrieval for question answering. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM; 2007. pp. 351–58.
6. Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J. 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, pp. 1247–50.
7. Bordes A, Usunier N, Chopra S, Weston J. Large-scale simple question answering with memory networks. 2015;36:2015. arXiv:1506.02075.
8. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. 2014. arXiv preprint arXiv:1406.1078.
9. Dai Z, Li L, Xu W. Cfo: Conditional focused neural question answering with large-scale knowledge bases. 2016. arXiv preprint arXiv:1606.01994.
10. Damljanovic D, Agatonovic M, Cunningham H. Natural language interfaces to ontologies: combining syntactic analysis and ontology-based lookup through the user interaction. In: Extended semantic web conference. Springer; 2010. pp. 106–120.
11. Dong L, Lapata M. Language to logical form with neural attention (2016). arXiv preprint arXiv:1601.01280.
12. Elman JL. Finding structure in time. Cogn Sci. 1990;14(2):179–21111.
13. Finkel JR, Grenager T, Manning CD. Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd annual meeting on association for computational linguistics. Association for computational linguistics; 2005. pp. 363–70.
14. Freitas A, Oliveira JG, O'Riain S, Curry E, Da Silva JCP, Treo: best-effort natural language queries over linked data. In: International conference on application of natural language to information systems. Springer; 2011. pp. 286–89.
15. Golub D, He X. Character-level question answering with attention. 2016. arXiv preprint arXiv:1604.00727.
16. Graves A. Generating sequences with recurrent neural networks. 2013. arXiv preprint arXiv:1308.0850.
17. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.
18. Hoffart J, Suchanek FM, Berberich K, Weikum G. Yago2: a spatially and temporally enhanced knowledge base from wikipedia. Artif Intell. 2013;194:28–61.
19. Huang X, Zhang J, Li D, Li P. Knowledge graph embedding based question answering. In: Proceedings of the twelfth ACM international conference on web search and data mining; 2019. pp. 105–13.
20. Hughes M, Li I, Kotoulas S, Suzumura T. Medical text classification using convolutional neural networks. Stud Health Technol Inform. 2017;235:246–50.
21. İrsoy O, Cardie C. Opinion mining with deep recurrent neural networks. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), Doha, Qatar, October 2014. Association for Computational Linguistics. pp. 720–28.
22. Kaufmann E, Bernstein A, Fischer L. Nlp-reduce: A naive but domain independent natural language interface for querying ontologies. In: 4th European semantic web conference ESWC; 2007. pp. 1–2.
23. Kim Y. Convolutional neural networks for sentence classification. 2014. arXiv preprint arXiv:1408.5882.
24. Kwiatkowksi T, Zettlemoyer L, Goldwater S, Steedman M. Inducing probabilistic CCG grammars from logical form with higher-order unification. In: Proceedings of the 2010 conference on empirical methods in natural language processing; 2010, pp. 1223–33.
25. Lafferty J, McCallum A, Pereira FC. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the eighteenth international conference on machine learning, ICML '01. San Francisco: Morgan Kaufmann Publishers Inc.; 2001. pp 282–89.
26. Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, Hellmann S, Morsey M, Van Kleef P, Auer S, et al. Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. Semant Web. 2015;6(2):167–95.
27. Levenshtein VI. Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady; 1966, vol 10. pp. 707–10.
28. Lopez V, Uren V, Motta E, Pasin M. Aqualog: an ontology-driven question answering system for organizational semantic intranets. Web Semant Sci Serv Agents World Wide Web. 2007;5(2):72–105.
29. Lopez V, Uren V, Sabou MR, Motta E. Cross ontology query answering on the semantic web: an initial evaluation. In:

Proceedings of the fifth international conference on Knowledge capture. ACM; 2009, pp. 17–24.
30. Lukovnikov D, Fischer A, Lehmann J, Auer S. Neural network-based question answering over knowledge graphs on word and character level. In Proceedings of the 26th international conference on world wide web. International world wide web conferences steering committee; 2017. pp. 1211–20.
31. Pennington J, Socher R, Manning C. Glove: global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP); 2014, pp. 1532–43.
32. Reimers N, Gurevych I. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. 2017. arXiv preprint arXiv:1707.09861.
33. Suwanwiwat H, Blumenstein M, Pal U. A complete automatic short answer assessment system with student identification. In: 2015 13th international conference on document analysis and recognition (ICDAR). IEEE; 2015, pp. 611–15.
34. Vrandečić D. Wikidata: a new platform for collaborative data collection. In Proceedings of the 21st international conference on world wide web. ACM; 2012, pp. 1063–64.
35. Yin W, Yu M, Xiang B, Zhou B, Schütze H. Simple question answering by attentive convolutional neural network. 2016. arXiv preprint arXiv:1409.04730.
36. Yu L, Hermann KM, Blunsom P, Pulman S. Deep learning for answer sentence selection. 2014. arXiv preprint arXiv:1409.04731.
37. Zettlemoyer LS, Collins M. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. 2012. arXiv preprint arXiv:1409.04732.