Generating Natural Language Question-Answer Pairs from a Knowledge Graph Using a RNN Based Question Generation Model

Sathish Indurthi [*], Dinesh Raghu[2], Mitesh M. Khapra [†] and Sachindra Joshi[2]

[1]Samsung Electronics, DMC R&D
[2]IBM Research India
[3]Indian Institute of Technology Madras
s.indurthi@samsung.com
{diraghu1,jsachind}@in.ibm.com
miteshk@cse.iitm.ac.in

Abstract

In recent years, knowledge graphs such as Freebase that capture facts about entities and relationships between them have been used actively for answering factoid questions. In this paper, we explore the problem of automatically generating question answer pairs from a given knowledge graph. The generated question answer (QA) pairs can be used in several downstream applications. For example, they could be used for training better QA systems. To generate such QA pairs, we first extract a set of keywords from entities and relationships expressed in a triple stored in the knowledge graph. From each such set, we use a subset of keywords to generate a natural language question that has a unique answer. We treat this subset of keywords as a sequence and propose a sequence to sequence model using RNN to generate a natural language question from it. Our RNN based model generates QA pairs with an accuracy of 33.61 percent and performs 110.47 percent (relative) better than a state-of-the-art template based method for generating natural language question from keywords. We also do an extrinsic evaluation by using the generated QA pairs to train a QA system and observe that the F1-score of the QA system improves by 5.5 percent (relative) when using automatically generated QA pairs in addition to manually generated QA pairs available for training.

1 Introduction

Knowledge graphs store information about millions of *things* (or entities) and relationships between them. Freebase[1] is one such knowledge graph that describes and organizes more than 3 billion *facts* in a consistent ontology. Knowledge graphs usually capture relationships between different *things* that can be viewed as triples (for example, `CEO(Sundar Pichai, Google)`). Such triples are often referred to as facts and can be used for answering factoid questions. For example, the above triple can be used to answer the question *"Who is the CEO of Google ?"*. It is not surprising that knowledge graphs are increasingly used for building Question Answering systems (Ferrucci, 2012; Yahya et al., 2013; He et al., 2014; Zou et al., 2014).

In this paper, we focus on exploiting knowledge graphs for a related but different purpose. We propose that such triples or facts can be used for automatically generating Question Answer (QA) pairs. The generated QA pairs can then be used in certain downstream applications. For example, if some domain-specific knowledge graphs are available (such as History, Geography) then such QA pairs generated from them can be used for developing quiz systems for educational purposes.

We now formally define the problem and then illustrate it with the help of an example. Consider a triple consisting of a subject, predicate and object. Typically, the predicate has a domain (subject type) and a range (object type) associated with it. The predicate may have zero or more parents in the knowledge graph. For the sake of simplicity let us assume that the predicate has a single parent. We define a set consisting of the subject, predicate, object, domain, range and predicate parent. We propose an approach

[*]This work was done while the author was a part of IBM Research India

[†]This work was done while the author was a part of IBM Research India

[1]https://www.freebase.com/

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 376–385,
Valencia, Spain, April 3-7, 2017.

Predicate	CEO
Subject	Sundar Pichai
Object	Google
Parent Predicate	designation
Domain	person
Range	organization
Keywords	CEO, designation, Sundar Pichai, person, Google, organization

Table 1: An example set of keywords constructed from the triple *CEO(Sundar Pichai, Google)*

to generate natural language factoid questions using a subset of this set such that the answer to the question also lies in the set. Given the set of keywords, as shown in Table 1, we could generate the following QA pairs (keywords are italicized):

Q: What is the *designation* of *Sundar Pichai* at *Google*?
A: *CEO*
Q: Which *organization* is *Sundar Pichai* the *CEO* of?
A: *Google*

The above problem is similar to the problem of generating questions from Web queries (instead of entities and relations) which was first suggested by Lin (Lin, 2008). However, unlike existing works on query-to-questions which mainly rely on template based approaches, we formulate this as a sequence to sequence generation problem wherein the ordered set of keywords is an input sequence and the natural language question is the output sequence. We use a Recurrent Neural Network (RNN) (Werbos, 1990; Rumelhart et al., 1988) based model with Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) units to generate questions from the given set of keywords.

The input to our question generation model is a set of keywords extracted from triples in a knowledge graph. For this, it is important to first select a subset of triples from which interesting and meaningful questions can be constructed. For example, no interesting questions can be constructed from the triple *wikipage_page_ID(Google, 57570)* and hence we should eliminate such triples. Further, even for an interesting triple, it may be possible to use only certain subsets of keywords to construct a meaningful question. For example, for the set of keywords shown in Table 1, it is not possible to use the subset {*person, designation*} to form an interesting question. Hence, we need to automatically identify the right set of keywords that should be used to form the question such that the answer also lies in the set. In addition to the question generation model, we also propose a method for extracting a meaningful subset of keywords from the triples represented in the knowledge graph.

While our goal in this paper is to generate a set of question answer pairs for a given entity in a knowledge graph, we train the RNN model for generating natural language questions from a sequence of keywords using an open domain Community Question Answering (CQA) data. This ensures that the same trained RNN can be used with different knowledge graphs.

The main contributions of our work can be summarized as follows:

- We propose a method for extracting triples and keywords from a knowledge graph for constructing question keywords and answer pairs.
- We formulate the problem of generating natural language questions from keywords as a sequence to sequence learning problem that performs 110.47 % (relative) better than existing template based approaches.
- We train our model using 1M questions from WikiAnswers thereby ensuring that it is not tied to any specific knowledge graph.
- Finally, we show that appending the automatically generated QA pairs to existing training data for training a state of the art QA system (Jonathan Berant, 2014) improves the performance of the QA system by 5.5 percent (relative).

The remainder of this paper is organized as follows. In next section, we describe related work, followed by a description of our overall approach for extracting keywords from triples and generating natural language question answer pairs from them. We then describe the experiments performed to evaluate our system and then end with concluding remarks.

2 Related Work

There is only very recent work around generation of question answer pairs from knowledge graph (Seyler et al., 2015). On the other hand, there are several works around question generation that have been proposed in past with different motivations. We first present a brief overview of the question generation techniques proposed in the literature along with their limitations and then discuss the work around generation of questions answer pairs from knowledge graph.

A number of papers have looked at the problem of generating vocabulary questions using WordNet (Miller et al., 1990) and distributional similarity techniques (Brown et al., 2005; Heilman and Eskenazi, 2007). There are numerous works in automatic question generation from text. Many proposed methods are *syntax based* methods that use the parse structure of sentences, identify key phrases and apply some known transformation rules to create questions (Ali et al., 2010; Kalady et al., 2010; Varga, 2010). Mannem et al. (2010) further use semantic role labeling for transformation rules. There are also *template based* methods proposed where a question template is a predefined text with placeholder variables to be replaced with content from source text. Cai et al. (2006) propose an XML markup language that is used to manually create question templates. This is sensitive to the performance of syntactic and semantic parsing. Heilman and Smith (2010) use a rule based approach to transform a declarative sentence into several candidate questions and then rank them using a logistic regression model. These approaches involve creating templates manually and thus require huge manual work and have low recall.

A problem that has been studied recently and is similar to our problem of generating questions using knowledge graph is that of generating questions from Web queries. The motivation here is to automatically generate questions from queries for community-based question answering services such as Yahoo! Answers and WikiAnswers. The idea was first suggested by (Lin, 2008) and further developed by (Zhao et al., 2011) and (Zheng et al., 2011). Both of these approaches are template based approaches where the templates are learnt using a huge question corpus along with query logs. Dror et al. (2013) further proposed a learning to rank based method to obtain grammatically correct and diverse questions from a given query where the candidate questions are generated using the approach proposed by (Zhao et al., 2011). These approaches use millions of query question pairs to learn question templates and thus have better generalization performance compared to earlier methods where templates were learnt manually.

Recently, Seyler et al. (2015) proposed a method to generate natural language questions from knowledge graphs given a topic of interest. They also provide a method to estimate difficulty of generated questions. The generation of question is done by manually created template patterns and therefore is limited in application. In contrast we propose an RNN based method to learn generation of natural language questions from a set of keywords. The model can be trained using a dataset containing open domain keywords and question pairs.

3 Approach

In this section we propose an approach to generate Question Answer (QA) pairs for a given entity E. Let KG be the knowledge graph which contains information about various entities in the form of triples. A triple consists of a subject, a predicate and an object. Subjects and objects are nodes in the KG, which could represent a person, a place, an abstract concept or any physical entity. Predicates are edges in the KG. They define type of relationship between the subject and the object.

The framework to generate QA pairs consists two major modules. The first module, **Question Keywords and Answer Extractor**, is language independent and extracts required knowledge about the entity E from the KG. The second module is a language dependent **RNN based Natural Language Question Generator**. When fed with the information extracted from the first part it generates natural language QA pairs.

3.1 Question Keywords and Answer Extractor

Question keywords are keywords necessary to generate a question in natural language, or it could also be viewed as a concise representation of a natural language question. For example, to generate a QA pair for the entity *London*. We can generate a natural language question like *What is the capital city of United Kingdom?* with the keywords {*Capital, City, United Kingdom*}. Also

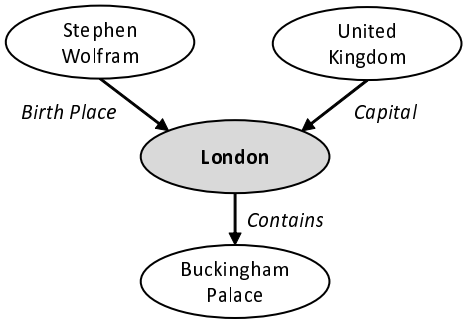

Figure 1: Triples with the entity *London* in a knowledge graph

since *London* is the answer to the above question, *({Capital, City, United Kingdom}, London)* together will form a Question Keyword and Answer (QKA) pair. One important note is that *Capital, City, United Kingdom* and *London* are the English labels of the node that represent these entities in the KG.

	Column A	*Column B*	*Column C*
Subject	United Kingdom	Stephen Wolfram	**London**
Domain	Country	Person	Location
Predicate	Capital	Birth Place	Contains
Object	**London**	**London**	Buckingham Palace
Range	City	Location	Location

Table 2: Examples of 5-tuples (subject, domain, predicate, object, range) for the entity *London*

In order to retrieve information about the given entity E, we need to first identify the node n that represents the entity E in the KG. One way to identify node n is to leverage the label (e.g, *rdfs:label*) property.

The next step is to retrieve all the neighbours of n. Let m_i be a neighbour of n in KG, connected by a predicate p_i. Here i is the index over all predicates whose subject or object is n. Figure 1 shows the entity *London* with three neighbours *United Kingdom*, *Stephen Wolfram* and *Buckingham Palace*. Each of these neighbours are related to London by a predicate. For example, *Stephen Wolfram* is related to *London* as it is his *Birth Place*.

Given a predicate p_i, let $sub(p_i)$ be the subject of p_i and $obj(p_i)$ be the object of p_i. A predicate is usually defined with a domain (subject type) and a range (object type) to provide better semantics. The domain and range defines the entity types that can be used as the subject and object of the predicate respectively. Let $domain(p_i)$ and $range(p_i)$ be the domain and range of p_i respectively. Let $\{sub(p_i), domain(p_i), p_i, obj(p_i), range(p_i)\}$ be the 5-tuple associated with every p_i. Some examples of 5-tuples are shown column wise in Table 2.

We now describe how QKA pairs are extracted from 5-tuple. Let Q_k be the question keywords set and A_k be the answer to the question to be generated using Q_k. (Q_k, A_k) together will form a QKA pair. In this work, we consider only a single 5-tuple to generate a QKA pair. For example, we can generate QKA pair like *({Capital, City, United Kingdom}, London)* using *Column A* of Table 2. But we will not generate QKA pair like *({Capital, City, United Kingdom, Birth Place, Stephen Wolfram}, London)* using both *Column A & B* of Table 2.

We use the following rules to generate QKA pairs from 5-tuples.

1. **Unique Forward Relation :** If p_i is unique for $sub(p_i)$ in KG, then Q_k will include $sub(p_i)$, p_i and $range(p_i)$. A_k will be $obj(p_i)$. If p_i is not unique for $sub(p_i)$, then there could be multiple possible answers to the generated question including $obj(p_i)$, and therefore we do not generate such a QKA pair. When this is applied to *Column A* of Table 2, we generate *({Capital, City, United Kingdom}, London)* as a QKA pair. There is no QKA pair generated for *Column C* using this rule as London contains many locations like Buckingham Palace, City of Westminster, etc.

2. **Unique Reverse Relation :** If p_i is unique for $obj(p_i)$ in KG, then Q_k will include $obj(p_i)$, p_i and $domain(p_i)$. A_k is $sub(p_i)$. Similar to unique forward relation, this rule can be applied to *Column A* of Table 2 and cannot be applied to *Column B & C*.

3.2 RNN based Natural Language Question Generator

In the previous sub-section, we proposed an approach for creating *question keywords* and *answer* pairs. Now we propose a model for generating natural language questions from a given set of question keywords. We treat the keywords, $QK = \{qk_1, \cdots, qk_m\}$, as an input sequence and the question, $Q = \{q_1, \cdots, q_l\}$, as the output sequence. This design choice of treating a set of keywords as a sequence and not as a bag of words

allows us to generate different semantically valid questions from the same set K based on the order of the words in the set. For example, given the question keywords, $QK = \{King, Sweden\}$ we can generate two semantically valid questions by changing the order of King and Sweden: (i) Who is the King of Sweden? and (ii) Does Sweden have a King?

We propose a Natural Language Question Generation (NLQG) model that first encodes the input sequence using some distributed representation and then decodes the output sequence from this encoded representation. Specifically, we use a RNN based encoder and decoder recently proposed for language processing tasks by number of groups (Cho et al., 2014; Sutskever et al., 2014). We now formally define the encoder and decoder models.

Let m be the number of keywords in the input sequence. We represent each keyword using a fixed size vector $x_i \in \Re^n$. The function of the encoder is to map this sequence of x_i's to a fixed size encoding. We use a RNN to compute h_m using the following recursive equation:

$$h_i = \Phi(h_{i-1}, x_i), \tag{1}$$

where, $h_i \in \Re^n$ is the hidden representation at position i. h_m is the final encoded hidden state vector for this sequence. We use LSTM units (Hochreiter and Schmidhuber, 1997) as Φ for our implementation based on its recent success in language processing tasks (Bahdanau et al., 2015).

The function of the decoder is to compute the probability of the output sequence $Q = \{q_1, \cdots, q_l\}$ given the encoded vector h_m. Note that l is the length of the output sequence and may be different from m. This joint conditional probability of Q is decomposed into l conditional probabilities:

$$p(q_1, \cdots, q_l | h_m) = \prod_{j=1}^{l} p(q_j | \{q_1, \cdots, q_{j-1}\}, h_m). \tag{2}$$

Now we model $p(q_j | q_{<j}, h_m)$ at each position j by using a RNN decoder as follows:

$$p(q_j | q_{<j}, h_m) = \Theta(q_{j-1}, g_j, h_m), \tag{3}$$

where Θ is a non-linear function, that outputs the probability of q_j, and g_j is the hidden state of the decoder RNN.

To train this RNN model, we use a keyword sequence and question pairs generated from an open domain Community Question Answering website. We provide more details on how the data is created and used for training in the experiments section.

At runtime, every permutation of the question keywords QK extracted is fed as input to the trained RNN. We pick the question Q with the highest probability of generation across all permutations, as the question generated from the question keywords QK.

4 Experiments

In this section we perform experiments to demonstrate how the proposed approach outperforms the existing template based approach for generating questions from the keywords. We also evaluate the quality of the QA pairs generated from knowledge graph.

4.1 Datasets

For training the K2Q-RNN model we require a set of keywords and question pairs. We use a large collection of open-domain questions available from WikiAnswers dataset[2]. This dataset has around 20M questions. We randomly selected 1M questions from this corpus for training and 5k questions for testing (the maximum length of a question was restricted to 50 words). We extract keywords from the selected questions by retaining only Nouns, Verbs and Adjectives in the question. The parts of speech tags were identified using Stanford Tagger (Toutanova et al., 2003). We form an ordered sequence of keywords by retaining these extracted words in the same order in which they appear in the original question. This sequence of keywords along with the original question forms one input-output sequence pair for training.

4.2 Methods

We evaluate and compare the following methods:

K2Q-RNN: This is our approach proposed in the paper. For the encoder we use a bi-directional RNN containing one hidden layer of 1000 units. Each word in the input vocabulary is represented using a word vector which is randomly initialized and learnt during the training process. The decoder also contains one hidden layer comprising of 1000 units. At the output layer of the decoder a softmax function gives the distribution over the entire target vocabulary. We use the top 30,000

[2] Available at http://knowitall.cs.washington.edu/oqa/data/wikianswers/

most frequent words in the 1M training questions as the target vocabulary. If any sequence contains a word not belonging to this list then that word is mapped to a special token ([UNK]) that is also considered a part of the output vocabulary. We use a mini batch stochastic gradient descent algorithm together with Adadelta (Zeiler, 2012) to train our model. We used a mini-batch size of 50 and trained the model for 10 epochs. We used the beam search with the beam size to 12 to generate the question that approximately maximizes conditional probability defined in Equation 2.

K2Q-PBSMT: As mentioned earlier, we treat the problem of generating questions from keywords as a sequence to sequence translation problem. A Phrase Based Machine Translation System (PBSMT) can also be employed for this task by considering that the keyword sequences belong to a source language and the question sequences belong to a target language. We compare our approach with a standard phrase-based MT system, MOSES (Koehn et al., 2007) trained using the same 1M sequence pairs constructed from the WikiAnswers dataset. We used a 5-gram language model trained on the 1M target question sequences and tuned the parameters of the decoder using 1000 held-out sequence (these were held out from the 1M training pairs).

K2Q-Template: For template based approach we use the method proposed by (Zhao et al., 2011) along with the Word2Vec (Mikolov et al., 2015) ranking as proposed by (Raghu et al., 2015). The Word2Vec ranking provides better generalization than the ranking proposed by (Zhao et al., 2011). We learn the templates using the same 1M training pairs extracted from WikiAnswers.

4.3 Evaluation metrics

We evaluate the performance of K2Q RNN with other baselines to compare the K2Q approaches, we use BLEU score (Papineni et al., 2002) between the generated question and the reference question. BLEU score is typically used in evaluating the performance of MT systems and captures the average n-gram overlap between the generated sequence and the reference sequence. We consider n-grams upto length 4. BLEU score does not capture the true performance of the system. For example, if the trained model simply reproduces all keywords in the generated question then also the unigram overlap will be high resulting in a higher BLEU score. Further, we had only one reference question (ground truth) per test instance which is not sufficient to capture the different ways of expressing the question. In this case, BLEU score will be unnecessarily harsh on the model even if it generates a valid paraphrase of the reference question. To account for this we also perform a manual evaluation. We show the generated output to four human annotator and ask him/her to assign following ratings to the generated question, Rating 4 : Perfect without error, Rating 3 : Good with one error, missing/addition of article or preposition, but still meaningful, Rating 2 : Many errors, Rating 1 : Failure.

Method	BLEU Score	Human Judgment accuracy (%)
K2Q-Template	25.58	28.57
K2Q-PBSMT	50.90	44.29
K2Q-RNN	50.14	60.13

Table 3: Automatic Evaluation (column 2): The BLEU scores of generated questions for the test set. Human Evaluation (Column 3): Percentage of perfect questions generated by *K2Q-Template, K2Q-PBSMT, and K2Q-RNN*

4.4 Results

4.4.1 RNN based Natural Language Question Generator

We first evaluate the performance of K2Q approaches using 5000 test instances from the WikiAnswers dataset. We extract the keyword sequence from these test questions using the same method described above. We compute the BLEU score by comparing the generated question with the original question. The results are presented in Table 3. Both K2Q-RNN and K2Q-PBSMT clearly outperform the template based method which shows that there is merit in formulating this problem as a sequence to sequence learning problem. To be sure that the results are not misleading due to some of the drawbacks of BLEU score as described earlier, we also do a manual evaluation. For this, we randomly selected 700 questions from the test set. We showed the questions generated by the three methods to different human annotators and asked them to assign a score of 1 to 4 to each question (based on the guidelines described earlier). The evaluators had no knowledge about the method used to generate each question shown to them. We only consider questions with rating 4 (perfect without any errors) for each

Ground truth	K2Q-PBSMT	K2Q-RNN
pitching in baseball ?	pitching in baseball ?	what is pitching in baseball ?
difference between mergeracqs and amalgamation ?	what is the difference between mergeracqs amalgamation ?	what is the difference between mergeracqs and amalgamation ?
did great britain control iraq ?	great britain control in iraq ?	how did the great britain control the iraq ?
what is the critical analysis of the poem a river ?	critical analysis of the poem a river ?	what is the most critical analysis of the poem a river ?
global warming affect population growth ?	global warming affect the population growth ?	can global warming affect the population growth ?

Table 4: Example questions from *Ground truth, K2Q-PBSMT and K2Q-RNN*

Entity	Keyword Query	Generated Question	Answer
Alan Turing	birth place alan turing	where is the birth place of alan turing ? (✓)	maida vale (✓)
	inventor lu decomposition	who was the inventor of lu decomposition ? (✓)	alan turing (✓)
	tv episodes alan turing	tv episodes of alan turing ? (✓)	dangerous knowledge (✓)
	author mathematical logic	what is the author of the mathematical logic ? (x)	alan turing (✓)
France	ioc code france	what is the ioc code for france ? (✓)	fr (✓)
	capital france	what is the capital of france ? (✓)	paris (✓)
	location lake annecy	what is the location of lake annecy ? (✓)	france (✓)
	albin haller country	is albin haller a country ? (x)	france (x)
Wimbledon	wimbledon first date occurrence	what was the wimbledon first date of occurrence ? (✓)	1877-07-09 (✓)
	current frequency wimbledon	what is the current frequency of wimbledon ? (✓)	yearly (✓)
	official website wimbledon	what is the official website for wimbledon ? (✓)	http://www.wimbledon.com/ (✓)

Table 5: Example question-answer pairs extracted for different entities by using *Freebase* and *K2Q-RNN*. Question-Answer pairs are considered correct if and only if both are marked with ✓by human judges.

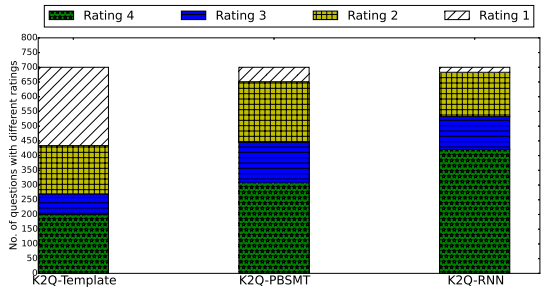

Figure 2: Ratings given by human judges for generated questions for *K2Q-Template, K2Q-PBSMT and K2Q-RNN* (Best viewed in color).

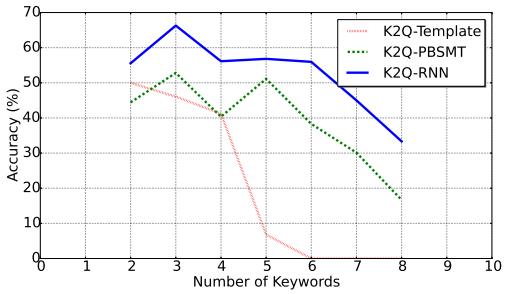

Figure 3: The plot shows the performance of 3 methods *K2Q-Template, K2Q-PBSMT, and K2Q-RNN* as a functions of number of keywords.

method and calculate the accuracy, shown in Table 3. Figure 2 shows the distribution of ratings assigned by the annotators. Once gain *K2Q-RNN* and *K2Q-PBSMT* outperform the template based approach. Further, the human evaluation shows *K2Q-RNN* performs better than *K2Q-PBSMT*. Table 4 shows example questions that may have a high BLEU score for *K2Q-PBSMT*, however the *K2Q-RNN* has a better human judgement.

Next, we also compare the performance of these methods for input keyword sequences of different lengths. For this, we consider all test instances having k keywords and mark the generated question as correct if it was given a rating of 4 by the human annotator. The results of this experiment are plotted in Figure 3 where the x-axis represents number of keywords and y-axis represents the percentage of test instances for which correct (rating 4) questions were generated. Once again we see that K2Q-RNN clearly outperforms the other methods at all input sequence sizes.

4.4.2 Generating Question-Answers pairs from Freebase

In this section we describe the performance of K2Q-RNN for generating QA pairs from a Knowledge Graph. For our evaluation purpose, we use *Freebase* as the Knowledge Graph. We randomly picked 27 Freebase entities of various types (person, location, organization, etc) and extracted all 5-tuples containing them. To create a diverse QA pairs we retained only two instances (5-tuples) for each predicate or relation type. Some predicates (like summary, quotations) have long text as their objects, some predicates (like Daylife Topic ID, Hero image ID) are difficult for annotator to validate. So, we filtered the list further by removing above mentioned predicates and generated a total of 485 QKA pairs. We manually evaluated these generated QA pairs and marked them as correct only if generated question along with the answer together convey the information represented

Train Sources	Train	Test	F1-score(%)
WQ	3778	2032	39.9
WQ+GQA	11334	2032	42.1
WQ+GT	11334	2032	43.6

Table 6: PARASEMPRE Evaluation: WQ=WebQuestions, GQA=Generated QA pairs from *SimpleQuestions* test dataset, GT=Ground truth data from *SimpleQuestions* test dataset.

in the 5-tuple. A few QA pairs were marked correct by the annotators, even though the question was not grammatically correct but convey the right intent. Some examples of such questions are *melting point of propyl alcohol?, stanford university student radio station?*. Overall, **33.61%** of the QA pairs generated by our method were annotated correct. Table 5 shows some correct and incorrect QA pairs generated by our method.

4.4.3 Extrinsic Evaluation

As an extrinsic evaluation of the quality of our QA generation model, we use QA pairs generated by our model to improve the performance of a state of the art QA system called PARASEMPRE (Jonathan Berant, 2014). PARASEMPRE is a semantic parser, which maps natural language questions to intermediate logical forms which in turn are used to answer the question. The standard training set used for training PARASEMPRE is a part of the *WebQuestions* and contains 3778 QA pairs. We appended this train with 7556 automatically generated QA pairs (resulting in tripling of the training set). Table 6 then compares the same system trained on the following different training sets: (i) Only Web Questions (WQ) dataset (ii) WQ + Generated Question Answers (GQA) and (iii) WQ + Ground Truth (GT) QA pairs. The GT QA pairs were obtained from the *SimpleQuestions* (Bordes et al., 2015) test data and have a one-to-one correspondence to the GQA data (hence the results are comparable). We see a relative improvement of 5.5% in the *F1-score* of the system by adding GQA. Further, the performance gains are comparable to those obtained by using GT QA pairs.

4.4.4 Error analysis

We inspected all the QA pairs generated by our method to identify some common mistakes. We found that most errors corresponded to (i) confusion between is/are and do/does (ii) incorrect use of determiners (missing articles, confusion between a/the and addition of extra articles). Another problem occurs when the extracted keyword sequence contains a stop word. This happens when dealing with triples such as ({also known as, Andre Agassi}, Agassi). Since, during training we retain only content words (nouns, adjectives, verbs) in the input sequence, the model fails to deal with such stop words at test time and simply produces unknown token (UNK) in the output. Another set of errors corresponds to mismatch between the subject type and question type. For example, we observed that in a few cases, the model incorrectly generates a *what question* instead of a *who question* when the answer type is a person.

5 Conclusions

In this paper we propose a method for generating QA pairs for an given entity using a knowledge graph. We also propose an RNN based approach for generating natural language questions from an input keyword sequence. The proposed method performs significantly better than previously proposed template based method. We also do an extrinsic evaluation to show that the generated QA pairs help in improving the performance of a downstream QA system. In future, we plan to extend this work to support predicates with stop words and support predicates in various tenses.

References

Husam Ali, Yllias Chali, and Sadid A Hasan. 2010. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 58–67.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Jonathan C Brown, Gwen A Frishkoff, and Maxine Eskenazi. 2005. Automatic question generation for vocabulary assessment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 819–826. Association for Computational Linguistics.

Zhiqiang Cai, Vasile Rus, Hyun-Jeong Joyce Kim, Suresh C Susarla, Pavan Karnam, and Arthur C

Graesser. 2006. Nlgml: A markup language for question generation. In *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, volume 2006, pages 2747–2752.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Gideon Dror, Yoelle Maarek, Avihai Mejer, and Idan Szpektor. 2013. From query to question in one click: suggesting synthetic questions to searchers. In *Proceedings of the 22nd international conference on World Wide Web*, pages 391–402. International World Wide Web Conferences Steering Committee.

David A Ferrucci. 2012. Introduction to this is watson. *IBM Journal of Research and Development*, 56(3.4):1–1.

Shizhu He, Kang Liu, Yuanzhe Zhang, Liheng Xu, and Jun Zhao. 2014. Question answering over linked data using first-order logic. In *Proceedings of Empirical Methods in Natural Language Processing*.

Michael Heilman and Maxine Eskenazi. 2007. Application of automatic thesaurus extraction for computer generation of vocabulary questions. In *SLaTE*, pages 65–68.

Michael Heilman and Noah A Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Percy Liang Jonathan Berant. 2014. Semantic parsing via paraphrasing. In *ACL*.

Saidalavi Kalady, Ajeesh Elikkottil, and Rajarshi Das. 2010. Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 1–10. questiongeneration. org.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chin-Yew Lin. 2008. Automatic question generation from queries. In *Workshop on the Question Generation Shared Task*, pages 156–164.

Prashanth Mannem, Rashmi Prasad, and Aravind Joshi. 2010. Question generation from paragraphs at upenn: Qgstec system description. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 84–91.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2015. Efficient estimation of word representations in vector space. In *ICLR*.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database*. *International journal of lexicography*, 3(4):235–244.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Dinesh Raghu, Sathish Indurthi, Jitendra Ajmera, and Sachindra Joshi. 2015. A statistical approach for non-sentential utterance resolution for interactive qa system. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 335.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of research. pages 696–699.

Dominic Seyler, Mohamed Yahya, and Klaus Berberich. 2015. Generating quiz questions from knowledge graphs. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 113–114. International World Wide Web Conferences Steering Committee.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*.

Andrea Varga. 2010. Le an ha 2010 wlv: A question generation system for the qgstec 2010 task b. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 80–83.

P.J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Oct.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1107–1116. ACM.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arxiv:*, 1212.5701.

Shiqi Zhao, Haifeng Wang, Chao Li, Ting Liu, and Yi Guan. 2011. Automatically generating questions from queries for community-based question answering. In *IJCNLP*, pages 929–937.

Zhicheng Zheng, Xiance Si, Edward Y Chang, and Xiaoyan Zhu. 2011. K2q: Generating natural language questions from keywords with user refinements. In *IJCNLP*, pages 947–955.

Lei Zou, Ruizhe Huang, Haixun Wang, Jeffer Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over rdf: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324. ACM.