The Value of Paraphrase for Knowledge Base Predicates

Bingcong Xue,[1] Sen Hu,[1] Lei Zou,[1,2] Jiashu Cheng[3]
[1]Peking University, China;
[2]Beijing Institute of Big Data Research, China;
[3]Culver Academies, USA
{xuebingcong, husen, zoulei}@pku.edu.cn, jiashu.cheng@culver.org

Abstract

Paraphrase, i.e., differing textual realizations of the same meaning, has proven useful for many natural language processing (NLP) applications. Collecting paraphrase for predicates in knowledge bases (KBs) is the key to comprehend the RDF triples in KBs. Existing works have published some paraphrase datasets automatically extracted from large corpora, but have too many redundant pairs or don't cover enough predicates, which cannot be improved by computer only and need the help of human beings. This paper shows a full process of collecting large-scale and high-quality paraphrase dictionaries for predicates in knowledge bases, which takes advantage of existing datasets and combines the technologies of machine mining and crowdsourcing. Our dataset comprises 2284 distinct predicates in DBpedia and 31130 paraphrase pairs in total, the quality of which is a great leap over previous works. Then it is demonstrated that such good paraphrase dictionaries can do great help to natural language processing tasks such as question answering and language generation. We also publish our own dictionary for further research.

1 Introduction

Paraphrase is the mapping from a text to a related one, where the two texts describe the same thing or have the same meaning. It can be understood as 'restate' based on semantic similarity and word ontology. Paraphrase generation is the process of collecting or generating object-paraphrase pairs for a given set of phrases, sentences or even passages. It often requires a subsequent verification stage to determine whether the two texts are paraphrastic, which is also called paraphrase recognition. However, due to the undemanding definition of paraphrase, this task cannot be perfectly finished by computer only. Instead it needs human's intuition, so it is often distributed to a large group of people and improved by human's intelligence, the process of which is known as crowdsourcing.

Crowdsourcing is the process of outsourcing a vast number of small, simple tasks to a distributed group of ordinary workers without specific skills. It is an effective way to solve computer-hard tasks and has many successful applications, such as entity resolution (Vesdapunt, Bellare, and Dalvi 2014; Wang, Xiao, and Lee 2015), image recognition (Welinder and Perona 2010), data cleaning (Park and Widom 2014; Wang et al. 2014) and knowledge construction (Amsterdamer et al. 2015), especially when the public crowdsourcing platforms, like Amazon Mechanical Turk (AMT)[1], CrowdFlower[2], and Upwork[3], have arisen and grown mature. In a crowdsourcing platform, requesters publish HITs (human intelligence tasks); Workers finish the tasks and return the results, and they will get reward when their answers are accepted. It is an easy part-time work pattern benefiting both workers and requesters. At the same time, however, as the tasks are distributed to a group of unskilled workers who may make mistakes and even give bad answers deliberately, the results obtained from crowdsourcing may have relatively low quality, which signifies delicate designs to control quality is of great importance in a crowdsourcing job. Besides, requesters often hope to spend less money to get better results in a shorter period of time. There is thus a compromise among quality, money and time. So in crowdsourcing tasks, appropriate designs often have a crucial impact on the results.

Knowledge bases (KBs) are gaining more attention for their wide use in many industrial fields. In a structured knowledge base, the Resource Description Framework (RDF) is the general framework for representing entities and their relations. Each RDF datum is stored as a triple composed of three elements, in the form of ⟨subject, predicate, object⟩. The predicate is often the key to comprehend the relation between entities (Zou and Özsu 2017). In a given knowledge base, predicates are limited and fixed, and sometimes hard for humans to understand, while natural language expressions are abundant for the same meaning. There is a need to map predicates of a KB to natural language expressions, i.e., collecting paraphrase for predicates in knowledge bases, which can be used in many NLP tasks such as semantic parsing, question answering (QA), machine translation and query generation. Some existing works focus on paraphrasing or mapping from predicates to natural language phrases and publish avaliable dictionar-

[1]https://www.mturk.com/
[2]http://www.crowdflower.com
[3]https://www.upwork.com

ies (Miller 1995; Nakashole, Weikum, and Suchanek 2012; Ganitkevitch, Durme, and Callison-Burch 2013). However, these dictionaries are mainly obtained automatically from large corpora and often have many redundant or wrong pairs, and they don't refer to or cover enough predicates. Applying these paraphrase dictionaries to NLP tasks may even degrade performances.

In this paper, we first investigate and analyze the inherent weaknesses of existing datasets and based on these works we propose a full procedure to collect a large-scale paraphrase dictionary for predicates in DBpedia, a widely-used knowledge base. Then we try to clean these data with the help of both machine algorithms and crowdsourcing, on the latter we focus on the control of quality, cost and latency and have delicate designs. After obtaining a large-scale and high-quality paraphrase dictionary, we apply it to two NLP tasks (question answering and question generation respectively) and gain satisfactory performance promotion, demonstrating the value of such good paraphrase dictionaries. We also publish our dataset for future research. More precisely, our contributions are as follows:

- We give a complete procedure of collecting paraphrase phrases for knowledge base predicates, which combines machine mining and crowdsourcing, and in this process we also optimize the design of crowdsourcing tasks.
- We release our dataset on github for further research[4]. Our dataset contains 2284 distinct predicates of DBpedia with 31130 satisfactory paraphrase pairs, which is a great leap over existing works.
- We introduce two NLP models for question answering (QA) and question generation (QG) respectively and show how paraphrase dictionaries can be used properly in such tasks. Extensive experiments are conducted and the results prove the value of such good paraphrase dictionaries for natural language processing tasks.

The rest of the paper is structured as follows. Section 2 introduces existing paraphrase datasets and analyses their inherent short comings. In Section 3, we show our processing procedure to get a satisfactory paraphrase dictionary as well as the design details of the crowdsourcing platform. Section 4 introduces two NLP models for QA and QG respectively and explains how paraphrase dictionaries can be used in such models. Then we perform extensive experiments with our new dictionary and present the results in Section 5. Section 6 concludes the study.

2 Related Dataset

WordNet (Miller 1995) is one of the most widely used lexical resources in computer science. It is an on-line lexical reference system inspired by psycholinguistic theories of human lexical memory. In this system, English nouns, verbs, adjectives and adverbs are organized into sets of synonyms, each representing one underlying lexical concept. And these synonyms are linked with different relations, including hypernym, antonymy and meronymy. It involves 155,287 words, which are arranged into 117,659 synonyms. But this dictionary is limited to single words and does not contain phrases or patterns. For example, in WordNet, word "spouse" and "partner" are in the same synset, based on which we can learn that "spouse" and "partner" express the same meaning in some contexts, but phrases like "be married with" that well paraphrase the word "spouse" are not included in. And this dictionary is a pure mapping from word to word and has nothing to do with predicates in knowledge bases. If phrases and relational patterns were avaliable, this dictionary could play bigger roles in natural language processing tasks.

Table 1: Some samples for predicate "award" in Patty.

Paraphrase	Score
nominate for	0.10964912280701754
finish in	0.07655502392344497
then sign by	0.05481283422459893
lead be	0.03374233128834356
when join [[det]]	0.02298221614227086
formerly play at	0.02127659574468085
be bury [[con]]	0.059907834101382486

Nakashole, Weikum, and Suchanek noticed the deficiency of phrases and patterns in WordNet and therefore created Patty (2012), a large resource for textual patterns that denote binary relations. It focuses on the compiling of binary relational patterns(i.e., phrases) between entities from a corpus and then builds a WordNet-style taxonomy for the binary relations. They obtain 350,569 pattern synsets on the Wikipedia corpus and then map these patterns to the predicates in DBpedia, a well-known knowledge base structured from Wikipedia (Bizer et al. 2009), which forms into a paraphrase dictionary with 225 distinct predicates and 127,811 corresponding paraphrase pairs. This work is a good start for automatically paraphrase construction for knowledge base predicates. However, their dataset contains only 225 predicates, which are far from enough in many NLP tasks. And each predicate has more than 500 paraphrase pairs in average, many of which are redundant or not good at all. Table 1 shows some paraphrase phrases for predicate "award", where many are far-fetched. Applying this dataset to NLP tasks may even degrade performances.

PPDB (Ganitkevitch, Durme, and Callison-Burch 2013; Pavlick et al. 2015) is a large-scale and multilingual paraphrase database automatically extracted from bilingual parallel corpora. It contains millions of paraphrases in 16 different languages and is arranged in three types: lexical (single word to single word), phrasal (multiword to single/multiword), and syntactic (paraphrase rules containing non-terminal symbols). The paraphrase pairs are ranked by score and the score is also used to divide the database into six sizes, where smaller size means higher precision. This database aims at handling language variability and unseen words in language processing and contains huge amounts of paraphrase pairs. However, there is a vital pity that this huge dataset is independent of knowledge bases and many paraphrase pairs are merely literally alike. Similar phrases

[4]https://github.com/xbc0112/paraphrase

Table 2: Statistics of related paraphrase datasets. (PPDB/W/S and PPDB/P/S stand for PPDB's English version of size S for word and phrase respectively.)

Dataset	Type	Scale
WordNet	word to word	155,287 words and 117,659 synsets
Patty	predicate to phrase	225 predicates and 127,811 pairs
PPDB/W/S	word to word	231,716 paraphrase pairs
PPDB/P/S	phrase to phrase	1,530,813 paraphrase pairs

Dataset	Features
Wordnet	no phrases, independent of predicates
Patty	lack predicates, unreliable paraphrase pairs
PPDB	pure paraphrase dictionary, too many word-like pairs

appear over and over again, which makes it look quite redundant.

Table 2 concludes the statistics of these datasets.

As the analysis above, some existing works devote to the construction of paraphrase pairs automatically and have released extensive databases. But these datasets generally have many redundant or wrong pairs and don't cover enough predicates. We take in their results and introduce machine algorithms and crowdsourcing for quality improvement and finally build a satisfactory paraphrase dictionary.

3 Paraphrase Acquisition

We make full use of the existing datasets and combine the technologies of machine mining and crowdsourcing to build a large-scale and high-quality paraphrase dictionary for predicates in DBpedia. This section shows the full process and the delicate designs of our crowdsourcing platform. In general, our procedure contains these steps: (1) Clean PPDB. (2) Merge Patty with PPDB. (3) Collect new frequent predicates in KBs. (4) Get paraphrase expressions for these new predicates from WordNet and PPDB. (5) Remove redundant and bad pairs by machine algorithms. (6) Paraphrase recognition by crowdsourcing. It can be broadly divided into machine processing stage and crowdsourcing stage, which will be explained in detail below.

3.1 Machine Processing Stage

We first choose PPDB's English packs for words and phrases of size S, which contains only the highest-scoring pairs. The reason we choose size S rather than larger ones is that the version of size S contains enough paraphrase pairs and larger ones are composed of too many redundant or bad pairs which will degrade quality. The two packs have about 230,000 and 1,220,000 paraphrase pairs respectively and many of these are literally alike. After eliminating redundant pairs by calculating edit distance between neighbouring strings and removing worthless phrases with odd characters or strings [5], we reduce the two packs into around 40,000 and 360,000 pairs separately.

Then we combine the contents in Patty and PPDB to build a large mapping from predicates to phrases. Patty is the mapping from predicate P to phrase A and PPDB is the mapping from phrase (or word) B to phrase (or word) C. To make full use of the paraphrases in PPDB and involve more paraphrase candidates for predicate P, we compare the distance between phrase A and phrase B in many ways, including conducting string matching, calculating the Levenshtein distance and measuring distance with Stanford's pretrained open-source word embedding[6]. When phrase A is considered equal to phrase B, phrase C is added to the paraphrase sets for predicate P. By this step we enlarge the candidate paraphrase sets for the 225 predicates that already reside in Patty, but potentially introduces more errors as well. Figure 1 shows this binding process.

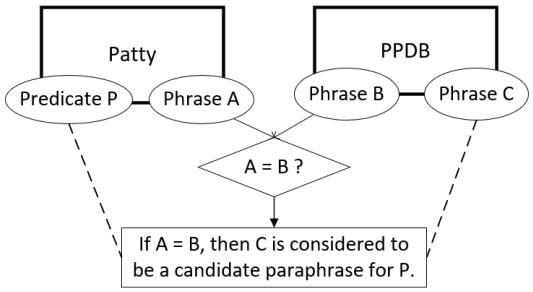

Figure 1: The binding process between Patty and PPDB.

In order to cover more predicates, we also start from the frequent predicates in DBpedia and analyze their denotations. Many predicates are natural words and stand for their literal meanings, which can get paraphrases from PPDB and WordNet by measuring the distance between the predicate and the phrases in existing datasets. Some don't have actual connotations literally but we can obtain their corresponding triples from DBpedia's resources (http://blog.dbpedia.org/), which can contribute to make out their meanings. We add explanations for these predicates by hand together with some corresponding triples, guaranteeing all the predicates shown to workers are comprehensible. Some predicates in KBs don't have meaningful triples and we directly omit them. By this step we expand our dictionary with more than 2000 predicates. Then we turn to crowdsourcing to get more paraphrases for these predicates, which can be further propagated with PPDB.

In this machine processing stage, we mainly take advantage of the existing datasets and by merging and culling we get a large dictionary composed of over 2000 predicates and plenty of candidate paraphrase pairs. Many bad paraphrase pairs are hidden here but cannot be removed appropriately by computer only, so we resort to crowdsourcing to do further paraphrase recognition.

3.2 Crowdsourcing Implementation

Crowdsourcing means handing out a vast amount of small, simple tasks to a distributed group of ordinary workers without specific skills. It is an effective way to solve computer-hard tasks and has been widely used these days. Quality, cost

[5] For example, some phrases in PPDB contain commas and consist of two segments such as "here , let me help you", which won't contribute to our work.

[6] http://nlp.stanford.edu/data/glove.840B.300d.zip

and latency control are three aspects that need to be taken into consideration in the design of crowdsourcing tasks and have huge influence on the results. And that is exactly what we focus on.

Instead of using public crowdsourcing platforms which have limited task formats and are hard to perform quality control, we build our own platform for better designs. In our platform, each HIT (human intelligence task) contains only one predicate and a number of candidate paraphrase words or phrases accessed from the machine processing stage. We want workers to give a score to determine the quality of each paraphrase for the predicate. In other words, we choose single choice in our tasks, which is the simplest task type in crowdsourcing task design and can easily attract enough workers in a short time. As (Tschirsich and Hintz 2013) has demonstrated that querying crowd-workers for a semantic similarity value on a multi-grade scale yields better results than directly asking for a binary classification, we give workers four choices (0,1,2,3) to evaluate the paraphrase pairs in our tasks and higher score means better paraphrase. No median choice forces the workers to give explicit preferences. Apart from single choices to determine the quality of the paraphrases, we also introduce an "open task" to encourage workers to write down more paraphrase phrases for each predicate optionally. Open tasks don't have a given range of answers, thus it is hard to determine or control the quality of the obtained answers from workers, which makes it not widely used in crowdsourcing tasks. But in our design, we merge single choices and blank fillings in each HIT, so we can determine the quality of the answers for the open tasks according to the confidence level of the workers calculated from the single choices, which gives a possible solution for this challenge. What's more, thorough instructions and heuristic scoring rules are presented to each worker in our platform. And for the obscure predicates, we give descriptions or corresponding triples from DBpedia to make them clearer to workers. In short, our design combines single choices and blank fillings, committed to provide a simple and definite platform for workers as well as convenient ways of quality control for requesters.

In term of quality control, we use a gold-injected method. A few golden paraphrases are hidden in the tasks and the workers' qualities can be computed according to their answers for the golden pairs. And in order to make the positions of the golden pairs more random, we build a hash map for these pairs, which holds back the probability that the regularities are founded out by workers. We also modify workers' quality parameters based on their answers' deviation from the majority of others'. It has been proved in experiments that this parameter can well represent workers' reliability and detect malicious workers. As we always assign each task to more than one worker (specifically three in our experiment), the final score of each paraphrase is computed by weighted majority voting after removing untrustworthy workers whose confidence level lower than a given threshold. As for cost and latency control, we use machine algorithms to remove similar, bad or redundant pairs as far as possible, so the number of pairs needed to be annotated by workers is reduced. We also deal with predicates with close meanings and transitivity is implemented conservatively here. Specifically, some predicates in KBs such as "designer" and "architect" are alike and their candidate paraphrases have much overlap, so we merge their paraphrases and they are only presented to workers once. What's more, we use accurate allocation algorithm to ensure all the tasks are marked evenly, i.e., HITs with the minimum number of labeled times are first showed to workers. Due to the simplicity of our task form, our tasks can easily attract enough workers in a short time.

We only handle the predicates with more than five candidate paraphrase phrases here. We limit the number of paraphrase phrases in each task less than 25 and divide our machine-processing data into near 2,000 HITs. If a predicate has more candidate phrases than 25, they will be scattered into several tasks. No more than 25 pairs in each task is to avoid workers' boredom. It takes one or two minutes to complete such a task. We publish these tasks on our platform and assign each task to three workers. All the tasks are finished within a week. Owing to the proper designs of our platform, we detect malicious workers in time and finally get satisfactory results. We remove bad paraphrase pairs based on workers' scoring and the remaining ones are ranked by their weighted scores. We also get extra 1,000 paraphrase pairs from reliable workers, which have been proved to be authentic and useful. Crowdsourcing does much help in our work and appropriate designs are of vital importance.

Table 3 shows some samples in our final dictionary.

Table 3: Some samples from our paraphrase dictionary

Predicate	Paraphrase	Score
birthPlace	be born in	100
birthPlace	native place	80
deathCause	die of	100
restingPlace	be buried in	90

4 Natural Language Processing Models

In this section we introduce two open-source NLP models for question answering and question generation respectively and show how paraphrase dictionaries can be used in such tasks.

4.1 QA Model: gAnswer

gAnswer (http://ganswer.gstore-pku.com/) is a state-of-the-art open-source question answering system over RDF. It introduces a semantic query graph to model the query intention of the natural language question in a structural way and reduce KBQA (question answering over knowledge base) to subgraph matching problem (Hu et al. 2018). We choose it to conduct our experiment on question answering.

The core of gAnswer lies in two aspects: how to build a semantic query graph Q^s for the natural language question N accurately and how to find matches of the query graph Q^s over RDF graph G efficiently. The node-first framework starts with finding nodes (entity/class phrases and wild-cards) and try to introduce edges to connect them to form a super

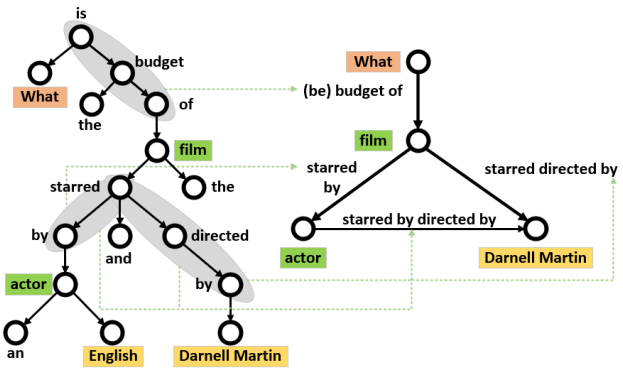

Figure 2: The super semantic query graph for a given question.

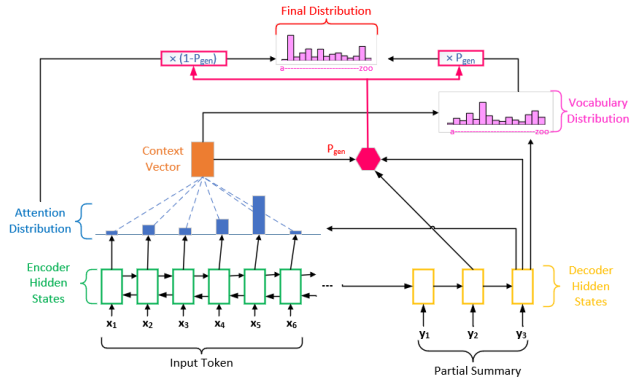

Figure 3: Pointer-generator model. It enhances the basic sequence-to-sequence attentional model with a generation probability P_{gen} for each decoder timestep, which is used to decide whether generating words from the vocabulary, or copying words from the source text.

query graph, which possibly has some uncertain or implicit relations (i.e., edges) and thus allows the structure ambiguity of query graph in the question understanding step. More precisely, the node-first framework contains these steps:

1) Node Recognition. They adopt a dictionary-based entity linking approach (Deng et al. 2015) to find entities and classes from the question sentence N and collect all wh-words and nouns that could not map to any entities and classes as wild-cards. All these are regarded as nodes in the semantic query graph Q^s.

2) Structure Construction. They first propose an assumption that two nodes v_1 and v_2 have a semantic relation if and only if there exists no other node v^* that occurs in the simple path between v_1 and v_2 of the dependency parse tree of question sentence N. Based on this assumption and the dependency tree from Stanford Parser (Marszalek-Kowalewska, Zaretskaya, and Soucek 2014), they introduce edges for the nodes recognized from last step. Figure 2 shows the super semantic query graph for the question sentence "What is the budget of the film starred by an English actor and directed by Darnell Martin", where each node and many edges have a corresponding label.

3) Phrases Mapping. In this step, they find candidate predicates and entities/classes in RDF graph for edges and nodes using offline dictionaries. Specifically, they adopt CrossWikis (Spitkovsky and Chang 2012) and Patty (Nakashole, Weikum, and Suchanek 2012) to map node and edge labels to entities and predicates respectively. Each node and edge may have more than one corresponding candidate with a confidence probability to be disambiguate in the matching stage.

4) Query Executing. This step searches for matches for the semantic query graph Q^s over RDF graph G, which addresses the ambiguities of phrases as well as obtaining the candidate answers. Instead of enumerating all spanning subgraphs of Q^s, they propose some efficient pruning strategies and a bottom-up algorithm to expand the partial structure step by step, which greatly speeds up the subgraph matching process.

In the whole process, phrases mapping is a key step linking the semantic query graph Q^s to RDF graph G. They use Patty (2012) to map natural language phrases to predicates, which contains few predicates and many redundant pairs. They argue that the failure of phrase mapping is the primary cause leading to the failure of some questions, the ratio of which is 31% (Hu et al. 2018). This shows the urgent demand and great effect of a large-scale and high-quality relation paraphrase dictionary. We later change their dictionary into ours and do extensive experiments, demonstrating the huge value of our paraphrase dictionary (see Section 5.2).

4.2 QG Model: Pointer-Generator Network

We choose a hot sequence-to-sequence model for question generation. Specifically, we adopt the model in (See, Liu, and Manning 2017), originally used for abstractive text summarization. In this work, they enhance the basic sequence-to-sequence attentional model (Nallapati et al. 2016) with a pointer network (Vinyals, Fortunato, and Jaitly 2015), which facilitates copying words from the source text and thus improves accuracy and handling of out-of-vocabulary (OOV) words, while retaining the ability to generate new words. They also introduce a *coverage vector* (Tu et al. 2016) from Neural Machine Translation, which is used to control coverage of the source document and avoid generating repetitive text. The whole model is showed in Figure 3.

In this network, they use articles as input and multi-sentence summaries as output. The tokens of the input article are fed one by one into the encoder (a single-layer bidirectional LSTM), producing a sequence of encoder hidden states, the decoder (a single-layer unidirectional LSTM) receives the word embedding of the previous word and has decoder state. The attention distribution is calculated as a probability distribution over the source words and used to produce a weighted sum of the encoder hidden states, known as the context vector. Then the context vector is concatenated with the decoder state and fed through two linear layers to produce a vocabulary distribution over all words. Apart from

this, they introduce the generation probability p_{gen} calculated from the context vector, the decoder state and the decoder input, which is used as a soft switch to choose between generating a word from the vocabulary distribution, or copying a word from the input sequence. This mechanism greatly intensifies the model's ability to handle OOV words. And in the calculation of the attention mechanism, they add a coverage vector summing the attention distributions over all previous decoder timesteps as extra input, which makes it easier for the attention mechanism to avoid repeatedly attending to the same locations and thus avoid generating repetitive text.

As the broad applicability of sequence-to-sequence model in NLP tasks and the contributions on reducing inaccuracies and repetition of their work, we modify its input and output for question generation task, where the input is standard semantic representation of a question such as SPARQLs or lambda calculus, and the output is a natural language question. The words produced by the primitive model are confined to the limited vocabulary, resulting in the lack of diversity of the generated questions, which is currently a main challenge in natural language generation tasks. That's exactly where paraphrase dictionaries can play an important role. We introduce our dictionary at the end of the model, selecting top expressions for the predicted predicates when mapping from id to word to compose the final questions, which helps to make the generated expressions more diverse. The implementation details can be found in Section 5.3.

5 Experiments

5.1 Datasets

We evaluate our dictionary on QALD (Question Answering over Linked Data), a series of open-domain question answering campaigns mainly based on DBpedia. Each piece of data in QALD contains a question and its corresponding SPARQLs and answers, which is convenient for testing. And many of these questions are not easy to answer.

We choose QALD6-QALD8 (Unger, Ngomo, and Cabrio 2016; Usbeck et al. 2017; 2018) to conduct experiments. The question numbers of these datasets can be found in Table 4[7].

5.2 Question Answering

We use gAnswer model to test our dictionary on question answering task. As its direct dependency on relation mapping dictionary, we change its old Patty dataset with about 130,000 paraphrase pairs into our new dictionary, which is one-fourth of it in size and contains more predicates. We evaluate their differences on the qald datasets and Table 4 shows the results. (Results with our new dictionary are denoted with '*', i.e., listed in the 3th, 5th, 7th, 9th lines.)

From Table 4 we can see that both recall and precision have a large promotion on all the datasets after substituting into our new dictionary. For example, the question "Who is the host of the BBC Wildlife Specials?" can be answered correctly with our dictionary but cannot with Patty, because a triple ⟨ presenter, host, 72 ⟩ resides in our dictionary and thus it successfully maps the relation phrase 'host' to the correct predicate 'presenter', while Patty doesn't have. The performances of the model vary across different datasets and it performs better on the training sets, which is mainly because the test datasets for the campaigns are harder and contain more sophisticated query structures. The model gets the highest precision on qald8-test dataset, but the recall is low. We analyze the questions in qald8-test and notice that many of these questions contain multiple or implicit relations or entities, which makes it harder for the model to understand the questions. But when the model successfully parses the question and gets the answer, it is often right.

Table 4: Evaluating QALD questions with gAnswer

	qald6-train	qald6-test	qald7-train	qald7-test	qald8-test
question	350	100	215	50	41
answered	196	43	121	23	15
answered*	201	47	133	23	15
right	106	17	75	11	10
right*	116	26	88	12	11
recall	0.303	0.170	0.349	0.220	0.244
recall*	0.331	0.260	0.409	0.240	0.268
precision	0.541	0.395	0.620	0.478	0.667
precision*	0.577	0.553	0.662	0.522	0.733

The dataset comprises corresponding SPARQLs of the questions as well, so we also compare the parsed SPARQLs with the two relation dictionaries. Since SPARQLs have multiple expressions for the same meaning and many predicates in DBpedia are quite similar (eg, "publish" and "publishes", "designer" and "architect"), evaluating their quality by machine algorithms can be cumbersome and implausible. So we analyze their differences by hand here. We ignore the trivial divergences such as replacements of synonymous predicates which come from the inherent imperfections of the knowledge base and different orders of the less important candidate SPARQLs. We also leave out the divergence when both give the wrong SPARQLs and answers. Thus we mainly focus on two kinds of changes: the first is that the model gets candidate SPARQLs with one dictionary but gets nothing with another, the other is that the SPARQLs obtained have important variance (especially when the predicates are different). We analyze the results of the qald7-train dataset, which is composed of 215 different questions. In this dataset, our dictionary outperforms Patty on 32 questions, 22 of which are the first change and 10 are the second, i.e., there are 22 questions that cannot get candidate SPARQLs by the model with Patty but obtain satisfactory SPARQLs with our dictionary, and 10 questions generate better candidate SPARQLs when switching into the new dictionary. Table 5 shows two examples reflecting the two kinds of changes respectively. Example 1 involves predicate "award". Though the Patty dictionary has 439 entries for this predicate, it includes so many wrong pairs (shown in Table 1) that it cannot get correct mapping for the phrase "win", thus gets nothing for this question. In example 2, Patty erroneously maps phrase "live in" to predicates "deathPlace" and "birthPlace", owing to its high scores for these two predicates and its lack for predicates like "residence". These two examples demonstrate that too many redundant pairs, lack of predicates and inaccurate confidence scores of the para-

[7]Every year's data contains a train and a test pack. As qald8-train merely merges the previous data, we omit it here.

Table 5: Examples of distinct SPARQL changes with different dictionaries. The old SPARQLs are those obtained with Patty and new SPARQLs are with our dictionary. Standard SPARQLs and answers are given in QALD datasets.

Example 1: Did Kaurismaki ever win the Grand Prix at Cannes?
Old SPARQLs: NULL
New SPARQLs:
[1]ask where ⟨Aki_Kaurismaki⟩ ⟨award⟩ ⟨Cannes⟩ ⟨Aki_Kaurismaki⟩ ⟨award⟩ ⟨Grand_PrixGSB⟩
[2]ask where ⟨Aki_Kaurismaki⟩ ⟨award⟩ ⟨Cannes⟩ ⟨Aki_Kaurismaki⟩ ⟨award⟩ ⟨M-1_Grand_Prix⟩
[3]ask where ⟨Aki_Kaurismaki⟩ ⟨award⟩ ⟨Cannes⟩ ⟨Aki_Kaurismaki⟩ ⟨award⟩ ⟨Pau_Grand_Prix⟩
Standard SPARQLs:
ask where ⟨http://dbpedia.org/resource/Aki_Kaurismaki⟩ ⟨http://dbpedia.org/ontology/award⟩ ⟨http://dbpedia.org/resource/Grand_Prix_(Cannes_Film_Festival)⟩
Answer: false
Example 2: How many people live in Poland?
Old SPARQLs:
[1]select COUNT(DISTINCT ?people) where ?people ⟨deathPlace⟩ ⟨Poland⟩
[2]select COUNT(DISTINCT ?people) where ?people ⟨birthPlace⟩ ⟨Poland⟩
[3]select COUNT(DISTINCT ?people) where ?people ⟨deathPlace⟩ ⟨Warsaw⟩
New SPARQLs:
[1]select COUNT(DISTINCT ?people) where ?people ⟨residence⟩ ⟨Poland⟩
[2]select COUNT(DISTINCT ?people) where ?people ⟨city⟩ ⟨Poland⟩
[3]select COUNT(DISTINCT ?people) where ?people ⟨location⟩ ⟨Poland⟩
Standard SPARQLs:
select distinct ?uri where ⟨http://dbpedia.org/resource/Poland⟩ ⟨http://dbpedia.org/ontology/populationTotal⟩ ?uri .
Answer: 38483957

phrase dictionary will degrade performance in Q/A systems.

As our dictionary covers all the predicates in Patty, there are no questions that can get SPARQLs with Patty but cannot with ours. And no SPARQLs are better using Patty because the confidence scores in our dictionary are more accurate. We also notice that for some questions the model gets the correct query graph but still derives wrong answers, which is due to the inherent imperfection of the model to handle complex aggregation operations. Besides, our dictionary is much smaller than Patty in size, leading to the fact that we can answer the questions faster.

5.3 Question Generation

We modify the pointer-generator network described in Section 4.2 for question generation task, which takes the SPARQLs of the QALD dataset as input and outputs corresponding natural language questions. We merge all the QALD datasets to form into a large one, composed of 737 distinct questions, from which we randomly choose 67 tuples for testing and others for training. After training for a whole day, this baseline model can generate appropriate questions for most of the SPARQLs, though some of them have syntax errors and are hard to comprehend. These are not the prime problems we focus on here. Instead, we want to introduce our paraphrase dictionary in this task to show its value on promoting diversity. Specifically, we add the paraphrase dictionary at the end of the model, devoted to randomly choosing a high-score paraphrase when mapping from the predicted id (obtained from the neural network) to the ultimate word. This is merely a preliminary attempt and doesn't change the structure of the model, only helping to increase lexical but not semantic diversity, which we believe can be designed more subtly later. But our dictionary does help to increase the diversity without reducing precision, especially when repeatedly forecasting the same word. Table 6 lists some examples where our paraphrase dictionary works. To measure the contributions on diversity of the dictionary, we adopt the *distinct-1* and *distinct-2* metrics from (Li et al. 2016), which are the number of distinct unigrams and bigrams in generated texts divided by total number of generated tokens respectively. Table 7 shows the results.

Table 6: Different expressions of the testing SPARQLs. The generated questions with and without paraphrase are denoted as n and o respectively and the standard ones are denoted as s.

o: Where does the deathPlace of Arabia?
n: Where does the assassination place of Arabia?
s: Where was JFK assassinated?
o: Give companies into are there in the advertising industry.
n: Give companies into are there in the advertising businesses.
s: Give me all companies in the advertising industry.
o: Where did the architect of the Eiffel Tower study?
n: Where did the architect who also design of the Eiffel Tower study?
s: Where did the architect of the Eiffel Tower study?
o: Give me all actors starring in movies by William Shatner.
n: Give me all actors have play as main roles in movies by William Shatner.
s: Give me all actors starring in movies directed by William Shatner.
o: Benicio starring movies produced by Benicio del Toro.
n: Benicio act the leading role in movies produced by Benicio del Toro.
s: Who is starring in Spanish movies produced by Benicio del Toro?

Table 7: Performance on diversity of the model with and without dictionary.

	unigrams	distinct unigrams	*distinct-1*	bigrams	distinct bigrams	*distinct-2*
no dictionary	453	226	0.499	386	313	0.811
with dictionary	466	235	0.504	399	325	0.815

From Table 6 we can see that our dictionary well paraphrases the predicted predicates, helping to avoid duplicate expressions for the same predicate (eg. 'starring' in the last two tuples). And it helps to increase the number of unigrams and bigrams as well as the *distinct-1* and *distinct-2* metrics, which demonstrates its value on promoting diversity. As the testing set is not large enough and the tokens don't recur frequently, our promotion seems not very conspicuous, which will be changed when the quantity of the testing entries increases substantially. Besides, we only conduct a minor change to the model and cannot alter the structure of the generated sentence, which is left to future researches. But there is no doubt that our paraphrase dictionary has enormous potential for promoting diversity in natural language generation tasks.

6 Conclusion

In this paper, we give a full process of collecting large-scale and high-quality paraphrase dictionaries for knowledge base predicates, which combines technologies of machine mining and crowdsourcing. We obtain a satisfactory dictionary for DBpedia and make two attempts on question answering and question generation tasks. The promotion of the performance demonstrates the value of such good paraphrase dictionaries in natural language processing tasks. The dictionary can be expanded to other knowledge bases and how to use paraphrase dictionaries dexterously in more tasks remains an open research question.

7 Acknowledgments

This work was supported by The National Key Research and Development Program of China under grant 2018YFB1003504 and NSFC under grant 61932001, 61961130390, 61622201 and 61532010. This work was also supported by Beijing Academy of Artificial Intelligence (BAAI). Lei Zou is the corresponding author of this paper.

References

Amsterdamer, Y.; Davidson, S. B.; Kukliansky, A.; Milo, T.; Novgorodov, S.; and Somech, A. 2015. Managing general and individual knowledge in crowd mining applications. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*.

Bizer, C.; Lehmann, J.; Kobilarov, G.; Auer, S.; Becker, C.; Cyganiak, R.; and Hellmann, S. 2009. Dbpedia - A crystallization point for the web of data. *J. Web Semant.* 7(3):154–165.

Deng, D.; Li, G.; Feng, J.; Duan, Y.; and Gong, Z. 2015. A unified framework for approximate dictionary-based entity extraction. *VLDB J.* 24(1):143–167.

Ganitkevitch, J.; Durme, B. V.; and Callison-Burch, C. 2013. PPDB: the paraphrase database. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, 758–764.

Hu, S.; Zou, L.; Yu, J. X.; Wang, H.; and Zhao, D. 2018. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Trans. Knowl. Data Eng.* 30(5):824–837.

Li, J.; Galley, M.; Brockett, C.; Gao, J.; and Dolan, B. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, 110–119.

Marszalek-Kowalewska, K.; Zaretskaya, A.; and Soucek, M. 2014. Stanford typed dependencies: Slavic languages application. In *Advances in Natural Language Processing - 9th International Conference on NLP, PolTAL 2014, Warsaw, Poland, September 17-19, 2014. Proceedings*, 151–163.

Miller, G. A. 1995. Wordnet: A lexical database for english. *Commun. ACM* 38(11):39–41.

Nakashole, N.; Weikum, G.; and Suchanek, F. M. 2012. PATTY: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, 1135–1145.

Nallapati, R.; Zhou, B.; dos Santos, C. N.; Gülçehre, Ç.; and Xiang, B. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, 280–290.

Park, H., and Widom, J. 2014. Crowdfill: collecting structured data from the crowd. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, 577–588.

Pavlick, E.; Rastogi, P.; Ganitkevitch, J.; Durme, B. V.; and Callison-Burch, C. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, 425–430.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, 1073–1083.

Spitkovsky, V. I., and Chang, A. X. 2012. A cross-lingual dictionary for english wikipedia concepts. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, 3168–3175.

Tschirsich, M., and Hintz, G. 2013. Leveraging crowdsourcing for paraphrase recognition. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013, August 8-9, 2013, Sofia, Bulgaria*, 205–213.

Tu, Z.; Lu, Z.; Liu, Y.; Liu, X.; and Li, H. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Unger, C.; Ngomo, A. N.; and Cabrio, E. 2016. 6th open challenge on question answering over linked data (QALD-6). In *Semantic Web Challenges - Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*, 171–177.

Usbeck, R.; Ngomo, A. N.; Haarmann, B.; Krithara, A.; Röder, M.; and Napolitano, G. 2017. 7th open challenge on question answering over linked data (QALD-7). In *Semantic Web Challenges - 4th SemWebEval Challenge at ESWC 2017, Portoroz, Slovenia, May 28 - June 1, 2017, Revised Selected Papers*, 59–69.

Usbeck, R.; Ngomo, A. N.; Conrads, F.; Röder, M.; and Napolitano, G. 2018. 8th challenge on question answering over linked data (QALD-8) (invited paper). In *Joint proceedings of the 4th Workshop on Semantic Deep Learning (SemDeep-4) and NLIWoD4: Natural Language Interfaces for the Web of Data (NLIWOD-4) and 9th Question Answering over Linked Data challenge (QALD-9) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, California, United States of America, October 8th - 9th, 2018.*, 51–57.

Vesdapunt, N.; Bellare, K.; and Dalvi, N. N. 2014. Crowdsourcing algorithms for entity resolution. *PVLDB* 7(12):1071–1082.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 2692–2700.

Wang, J.; Krishnan, S.; Franklin, M. J.; Goldberg, K.; Kraska, T.; and Milo, T. 2014. A sample-and-clean framework for fast and accurate query processing on dirty data. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, 469–480.

Wang, S.; Xiao, X.; and Lee, C. 2015. Crowd-based deduplication: An adaptive approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, 1263–1277.

Welinder, P., and Perona, P. 2010. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2010, San Francisco, CA, USA, 13-18 June, 2010*, 25–32.

Zou, L., and Özsu, M. T. 2017. Graph-based RDF data management. *Data Science and Engineering* 2(1):56–70.