Learning to Paraphrase for Question Answering

Learning to Paraphrase for Question Answering

Li Dong[†] and **Jonathan Mallinson**[†] and **Siva Reddy**[‡] and **Mirella Lapata**[†]
[†] ILCC, School of Informatics, University of Edinburgh
[‡] Computer Science Department, Stanford University
li.dong@ed.ac.uk, J.Mallinson@ed.ac.uk
sivar@stanford.edu, mlap@inf.ed.ac.uk

Abstract

Question answering (QA) systems are sensitive to the many different ways natural language expresses the same information need. In this paper we turn to paraphrases as a means of capturing this knowledge and present a general framework which learns felicitous paraphrases for various QA tasks. Our method is trained end-to-end using question-answer pairs as a supervision signal. A question and its paraphrases serve as input to a neural scoring model which assigns higher weights to linguistic expressions most likely to yield correct answers. We evaluate our approach on QA over Freebase and answer sentence selection. Experimental results on three datasets show that our framework consistently improves performance, achieving competitive results despite the use of simple QA models.

1 Introduction

Enabling computers to automatically answer questions posed in natural language on any domain or topic has been the focus of much research in recent years. Question answering (QA) is challenging due to the many different ways natural language expresses the same information need. As a result, small variations in semantically equivalent questions, may yield different answers. For example, a hypothetical QA system must recognize that the questions "*who created microsoft*" and "*who started microsoft*" have the same meaning and that they both convey the `founder` relation in order to retrieve the correct answer from a knowledge base.

Given the great variety of surface forms for semantically equivalent expressions, it should come as no surprise that previous work has investigated the use of paraphrases in relation to question answering. There have been three main strands of research. The first one applies paraphrasing to match natural language and logical forms in the context of semantic parsing. Berant and Liang (2014) use a template-based method to heuristically generate canonical text descriptions for candidate logical forms, and then compute paraphrase scores between the generated texts and input questions in order to rank the logical forms. Another strand of work uses paraphrases in the context of neural question answering models (Bordes et al., 2014a,b; Dong et al., 2015). These models are typically trained on question-answer pairs, and employ question paraphrases in a multi-task learning framework in an attempt to encourage the neural networks to output similar vector representations for the paraphrases.

The third strand of research uses paraphrases more directly. The idea is to paraphrase the question and then submit the rewritten version to a QA module. Various resources have been used to produce question paraphrases, such as rule-based machine translation (Duboue and Chu-Carroll, 2006), lexical and phrasal rules from the Paraphrase Database (Narayan et al., 2016), as well as rules mined from Wiktionary (Chen et al., 2016) and large-scale paraphrase corpora (Fader et al., 2013). A common problem with the generated paraphrases is that they often contain inappropriate candidates. Hence, treating all paraphrases as equally felicitous and using them to answer the question could degrade performance. To remedy this, a scoring model is often employed, however independently of the QA system used to find the answer (Duboue and Chu-Carroll, 2006; Narayan et al., 2016). Problematically, the separate paraphrase models used in previous work do not fully utilize the supervision signal from the training data, and as such cannot be properly tuned

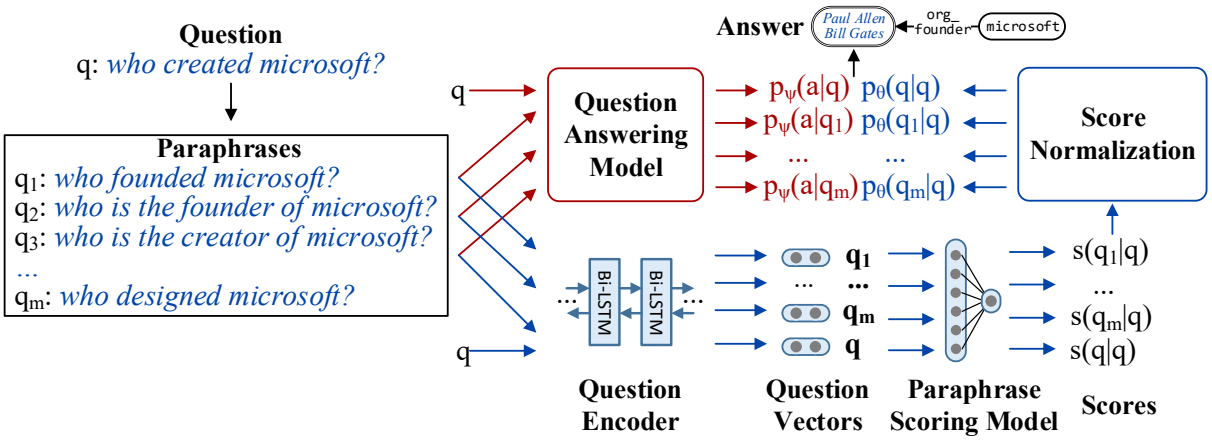

Figure 1: We use three different methods to generate candidate paraphrases for input q. The question and its paraphrases are fed into a neural model which scores how suitable they are. The scores are normalized and used to weight the results of the question answering model. The entire system is trained end-to-end using question-answer pairs as a supervision signal.

to the question answering tasks at hand. Based on the large variety of possible transformations that can generate paraphrases, it seems likely that the kinds of paraphrases that are useful would depend on the QA application of interest (Bhagat and Hovy, 2013). Fader et al. (2014) use features that are defined over the original question and its rewrites to score paraphrases. Examples include the pointwise mutual information of the rewrite rule, the paraphrase's score according to a language model, and POS tag features. In the context of semantic parsing, Chen et al. (2016) also use the ID of the rewrite rule as a feature. However, most of these features are not informative enough to model the quality of question paraphrases, or cannot easily generalize to unseen rewrite rules.

In this paper, we present a general framework for learning paraphrases for question answering tasks. Given a natural language question, our model estimates a probability distribution over candidate answers. We first generate paraphrases for the question, which can be obtained by one or several paraphrasing systems. A neural scoring model predicts the quality of the generated paraphrases, while learning to assign higher weights to those which are more likely to yield correct answers. The paraphrases and the original question are fed into a QA model that predicts a distribution over answers given the question. The entire system is trained end-to-end using question-answer pairs as a supervision signal. The framework is flexible, it does not rely on specific paraphrase or QA models. In fact, this plug-and-play functionality allows to learn specific paraphrases for different QA tasks and to explore the merits of different paraphrasing models for different applications.

We evaluate our approach on QA over Freebase and text-based answer sentence selection. We employ a range of paraphrase models based on the Paraphrase Database (PPDB; Pavlick et al. 2015), neural machine translation (Mallinson et al., 2016), and rules mined from the WikiAnswers corpus (Fader et al., 2014). Results on three datasets show that our framework consistently improves performance; it achieves state-of-the-art results on GraphQuestions and competitive performance on two additional benchmark datasets using simple QA models.

2 Problem Formulation

Let q denote a natural language question, and a its answer. Our aim is to estimate $p\left(a|q\right)$, the conditional probability of candidate answers given the question. We decompose $p\left(a|q\right)$ as:

$$p\left(a|q\right) = \sum_{q' \in H_q \cup \{q\}} \underbrace{p_\psi\left(a|q'\right)}_{\text{QA Model}} \underbrace{p_\theta\left(q'|q\right)}_{\text{Paraphrase Model}} \quad (1)$$

where H_q is the set of paraphrases for question q, ψ are the parameters of a QA model, and θ are the parameters of a paraphrase scoring model.

As shown in Figure 1, we first generate candidate paraphrases H_q for question q. Then, a neural scoring model predicts the quality of the generated paraphrases, and assigns higher weights to the paraphrases which are more likely to obtain

Input: what be the zip code of the largest car manufacturer	
what be the zip code of the largest vehicle manufacturer	PPDB
what be the zip code of the largest car producer	PPDB
what be the postal code of the biggest automobile manufacturer	NMT
what be the postcode of the biggest car manufacturer	NMT
what be the largest car manufacturer 's postal code	Rule
zip code of the largest car manufacturer	Rule

Table 1: Paraphrases obtained for an input question from different models (PPDB, NMT, Rule). Words are lowercased and stemmed.

the correct answers. These paraphrases and the original question simultaneously serve as input to a QA model that predicts a distribution over answers for a given question. Finally, the results of these two models are fused to predict the answer. In the following we will explain how $p(q'|q)$ and $p(a|q')$ are estimated.

2.1 Paraphrase Generation

As shown in Equation (1), the term $p(a|q)$ is the sum over q and its paraphrases H_q. Ideally, we would generate all the paraphrases of q. However, since this set could quickly become intractable, we restrict the number of candidate paraphrases to a manageable size. In order to increase the coverage and diversity of paraphrases, we employ three methods based on: (1) lexical and phrasal rules from the Paraphrase Database (Pavlick et al., 2015); (2) neural machine translation models (Sutskever et al., 2014; Bahdanau et al., 2015); and (3) paraphrase rules mined from clusters of related questions (Fader et al., 2014). We briefly describe these models below, however, there is nothing inherent in our framework that is specific to these, any other paraphrase generator could be used instead.

2.1.1 PPDB-based Generation

Bilingual pivoting (Bannard and Callison-Burch, 2005) is one of the most well-known approaches to paraphrasing; it uses bilingual parallel corpora to learn paraphrases based on techniques from phrase-based statistical machine translation (SMT, Koehn et al. 2003). The intuition is that two English strings that translate to the same foreign string can be assumed to have the same meaning. The method first extracts a bilingual phrase table and then obtains English paraphrases by pivoting through foreign language phrases.

Drawing inspiration from syntax-based SMT, Callison-Burch (2008) and Ganitkevitch et al. (2011) extended this idea to syntactic paraphrases,

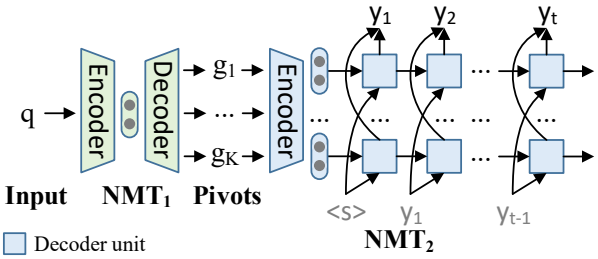

Figure 2: Overview of NMT-based paraphrase generation. NMT_1 (green) translates question q into pivots $g_1 \ldots g_K$ which are then back-translated by NMT_2 (blue) where K decoders jointly predict tokens at each time step, rather than only conditioning on one pivot and independently predicting outputs.

leading to the creation of PPDB (Ganitkevitch et al., 2013), a large-scale paraphrase database containing over a billion of paraphrase pairs in 24 different languages. Pavlick et al. (2015) further used a supervised model to automatically label paraphrase pairs with entailment relationships based on natural logic (MacCartney, 2009). In our work, we employ bidirectionally entailing rules from PPDB. Specifically, we focus on lexical (single word) and phrasal (multiword) rules which we use to paraphrase questions by replacing words and phrases in them. An example is shown in Table 1 where we substitute *car* with *vehicle* and *manufacturer* with *producer*.

2.1.2 NMT-based Generation

Mallinson et al. (2016) revisit bilingual pivoting in the context of neural machine translation (NMT, Sutskever et al. 2014; Bahdanau et al. 2015) and present a paraphrasing model based on neural networks. At its core, NMT is trained end-to-end to maximize the conditional probability of a correct translation given a source sentence, using a bilingual corpus. Paraphrases can be obtained by translating an English string into a foreign language and then back-translating it into English. NMT-based pivoting models offer advantages over conventional methods such as the ability to learn continuous representations and to consider wider context while paraphrasing.

In our work, we select German as our pivot following Mallinson et al. (2016) who show that it outperforms other languages in a wide range of paraphrasing experiments, and pretrain two NMT systems, English-to-German (EN-DE) and

Source	Target
the average size of __	what be __ average size
__ be locate on which continent	what continent be __ a part of
language speak in __	what be the official language of __
what be the money in __	what currency do __ use

Table 2: Examples of rules used in the rule-based paraphrase generator.

German-to-English (DE-EN). A naive implementation would translate a question to a German string and then back-translate it to English. However, using only one pivot can lead to inaccuracies as it places too much faith on a single translation which may be wrong. Instead, we translate from multiple pivot sentences (Mallinson et al., 2016). As shown in Figure 2, question q is translated to K-best German pivots, $\mathcal{G}_q = \{g_1, \ldots, g_K\}$. The probability of generating paraphrase $q' = y_1 \ldots y_{|q'|}$ is decomposed as:

$$\begin{aligned} p\left(q'|\mathcal{G}_q\right) &= \prod_{t=1}^{|q'|} p\left(y_t|y_{<t}, \mathcal{G}_q\right) \\ &= \prod_{t=1}^{|q'|} \sum_{k=1}^{K} p\left(g_k|q\right) p\left(y_t|y_{<t}, g_k\right) \end{aligned} \tag{2}$$

where $y_{<t} = y_1, \ldots, y_{t-1}$, and $|q'|$ is the length of q'. Probabilities $p\left(g_k|q\right)$ and $p\left(y_t|y_{<t}, g_k\right)$ are computed by the EN-DE and DE-EN models, respectively. We use beam search to decode tokens by conditioning on multiple pivoting sentences. The results with the best decoding scores are considered candidate paraphrases. Examples of NMT paraphrases are shown in Table 1.

Compared to PPDB, NMT-based paraphrases are syntax-agnostic, operating on the surface level without knowledge of any underlying grammar. Furthermore, paraphrase rules are captured implicitly and cannot be easily extracted, e.g., from a phrase table. As mentioned earlier, the NMT-based approach has the potential of performing major rewrites as paraphrases are generated while considering wider contextual information, whereas PPDB paraphrases are more local, and mainly handle lexical variation.

2.1.3 Rule-Based Generation

Our third paraphrase generation approach uses rules mined from the WikiAnswers corpus (Fader et al., 2014) which contains more than 30 million question clusters labeled as paraphrases by WikiAnswers[1] users. This corpus is a large resource (the average cluster size is 25), but is relatively noisy due to its collaborative nature – 45% of question pairs are merely related rather than genuine paraphrases. We therefore followed the method proposed in (Fader et al., 2013) to harvest paraphrase rules from the corpus. We first extracted question templates (i.e., questions with at most one wild-card) that appear in at least ten clusters. Any two templates co-occurring (more than five times) in the same cluster and with the same arguments were deemed paraphrases. Table 2 shows examples of rules extracted from the corpus. During paraphrase generation, we consider substrings of the input question as arguments, and match them with the mined template pairs. For example, the stemmed input question in Table 1 can be paraphrased using the rules ("*what be the zip code of __*", "*what be __ 's postal code*") and ("*what be the zip code of __*", "*zip code of __*"). If no exact match is found, we perform fuzzy matching by ignoring stop words in the question and templates.

2.2 Paraphrase Scoring

Recall from Equation (1) that $p_\theta\left(q'|q\right)$ scores the generated paraphrases $q' \in H_q \cup \{q\}$. We estimate $p_\theta\left(q'|q\right)$ using neural networks given their successful application to paraphrase identification tasks (Socher et al., 2011; Yin and Schütze, 2015; He et al., 2015). Specifically, the input question and its paraphrases are encoded as vectors. Then, we employ a neural network to obtain the score $s\left(q'|q\right)$ which after normalization becomes the probability $p_\theta\left(q'|q\right)$.

Encoding Let $q = q_1 \ldots q_{|q|}$ denote an input question. Every word is initially mapped to a d-dimensional vector. In other words, vector $\mathbf{q}_t$ is computed via $\mathbf{q}_t = \mathbf{W}_q \mathbf{e}\left(q_t\right)$, where $\mathbf{W}_q \in \mathbb{R}^{d \times |\mathcal{V}|}$ is a word embedding matrix, $|\mathcal{V}|$ is the vocabulary size, and $\mathbf{e}\left(q_t\right)$ is a one-hot vector. Next, we use a bi-directional recurrent neural network with long short-term memory units (LSTM, Hochreiter and Schmidhuber 1997) as the question encoder, which is shared by the input questions and their paraphrases. The encoder recursively processes tokens one by one, and uses the encoded vectors to represent questions. We compute the hidden vectors at the t-th time step via:

[1] wiki.answers.com

$$\overrightarrow{\mathbf{h}}_t = \text{LSTM}\left(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{q}_t\right), t = 1, \ldots, |q|$$
$$\overleftarrow{\mathbf{h}}_t = \text{LSTM}\left(\overleftarrow{\mathbf{h}}_{t+1}, \mathbf{q}_t\right), t = |q|, \ldots, 1 \quad (3)$$

where $\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t \in \mathbb{R}^n$. In this work we follow the LSTM function described in Pham et al. (2014). The representation of q is obtained by:

$$\mathbf{q} = \left[\overrightarrow{\mathbf{h}}_{|q|}, \overleftarrow{\mathbf{h}}_1\right] \quad (4)$$

where $[\cdot, \cdot]$ denotes concatenation, and $\mathbf{q} \in \mathbb{R}^{2n}$.

Scoring After obtaining vector representations for q and q', we compute the score $s(q'|q)$ via:

$$s\left(q'|q\right) = \mathbf{w}_s \cdot \left[\mathbf{q}, \mathbf{q}', \mathbf{q} \odot \mathbf{q}'\right] + b_s \quad (5)$$

where $\mathbf{w}_s \in \mathbb{R}^{6n}$ is a parameter vector, $[\cdot, \cdot, \cdot]$ denotes concatenation, $\odot$ is element-wise multiplication, and b_s is the bias. Alternative ways to compute $s(q'|q)$ such as dot product or with a bilinear term were not empirically better than Equation (5) and we omit them from further discussion.

Normalization For paraphrases $q' \in H_q \cup \{q\}$, the probability $p_\theta(q'|q)$ is computed via:

$$p_\theta\left(q'|q\right) = \frac{\exp\{s\left(q'|q\right)\}}{\sum_{r \in H_q \cup \{q\}} \exp\{s\left(r|q\right)\}} \quad (6)$$

where the paraphrase scores are normalized over the set $H_q \cup \{q\}$.

2.3 QA Models

The framework defined in Equation (1) is relatively flexible with respect to the QA model being employed as long as it can predict $p_\psi(a|q')$. We illustrate this by performing experiments across different tasks and describe below the models used for these tasks.

Knowledge Base QA In our first task we use the Freebase knowledge base to answer questions. Query graphs for the questions typically contain more than one predicate. For example, to answer the question "*who is the ceo of microsoft in 2008*", we need to use one relation to query "*ceo of microsoft*" and another relation for the constraint "*in 2008*". For this task, we employ the SIMPLEGRAPH model described in Reddy et al. (2016, 2017), and follow their training protocol and feature design. In brief, their method uses rules to convert questions to ungrounded logical forms, which are subsequently matched against Freebase subgraphs. The QA model learns from question-answer pairs: it extracts features for pairs of questions and Freebase subgraphs, and uses a logistic regression classifier to predict the probability that a candidate answer is correct. We perform entity linking using the Freebasee/KG API on the original question (Reddy et al., 2016, 2017), and generate candidate Freebase subgraphs. The QA model estimates how likely it is for a subgraph to yield the correct answer.

Answer Sentence Selection Given a question and a collection of relevant sentences, the goal of this task is to select sentences which contain an answer to the question. The assumption is that correct answer sentences have high semantic similarity to the questions (Yu et al., 2014; Yang et al., 2015; Miao et al., 2016). We use two bidirectional recurrent neural networks (BILSTM) to separately encode questions and answer sentences to vectors (Equation (4)). Similarity scores are computed as shown in Equation (5), and then squashed to $(0, 1)$ by a sigmoid function in order to predict $p_\psi(a|q')$.

2.4 Training and Inference

We use a log-likelihood objective for training, which maximizes the likelihood of the correct answer given a question:

$$\text{maximize} \sum_{(q,a) \in \mathcal{D}} \log p\left(a|q\right) \quad (7)$$

where $\mathcal{D}$ is the set of all question-answer training pairs, and $p(a|q)$ is computed as shown in Equation (1). For the knowledge base QA task, we predict how likely it is that a subgraph obtains the correct answer, and the answers of some candidate subgraphs are partially correct. So, we use the binary cross entropy between the candidate subgraph's F1 score and the prediction as the objective function. The RMSProp algorithm (Tieleman and Hinton, 2012) is employed to solve this non-convex optimization problem. Moreover, dropout is used for regularizing the recurrent neural networks (Pham et al., 2014).

At test time, we generate paraphrases for the question q, and then predict the answer by:

$$\hat{a} = \arg\max_{a' \in \mathcal{C}_q} p\left(a'|q\right) \quad (8)$$

where $\mathcal{C}_q$ is the set of candidate answers, and $p(a'|q)$ is computed as shown in Equation (1).

3 Experiments

We compared our model which we call PARA4QA (as shorthand for learning to paraphrase for question answering) against multiple previous systems on three datasets. In the following we introduce these datasets, provide implementation details for our model, describe the systems used for comparison, and present our results.

3.1 Datasets

Our model was trained on three datasets, representative of different types of QA tasks. The first two datasets focus on question answering over a structured knowledge base, whereas the third one is specific to answer sentence selection.

WEBQUESTIONS This dataset (Berant et al., 2013) contains 3,778 training instances and 2,032 test instances. Questions were collected by querying the Google Suggest API. A breadth-first search beginning with *wh-* was conducted and the answers were crowd-sourced using Freebase as the backend knowledge base.

GRAPHQUESTIONS The dataset (Su et al., 2016) contains 5,166 question-answer pairs (evenly split into a training and a test set). It was created by asking crowd workers to paraphrase 500 Freebase graph queries in natural language.

WIKIQA This dataset (Yang et al., 2015) has 3,047 questions sampled from Bing query logs. The questions are associated with 29,258 candidate answer sentences, 1,473 of which contain the correct answers to the questions.

3.2 Implementation Details

Paraphrase Generation Candidate paraphrases were stemmed (Minnen et al., 2001) and lowercased. We discarded duplicate or trivial paraphrases which only rewrite stop words or punctuation. For the NMT model, we followed the implementation[2] and settings described in Mallinson et al. (2016), and used English↔German as the language pair. The system was trained on data released as part of the WMT15 shared translation task (4.2 million sentence pairs). We also had access to back-translated monolingual training data (Sennrich et al., 2016a). Rare words were split into subword units (Sennrich et al., 2016b) to handle out-of-vocabulary words in questions. We used the top 15 decoding results as candidate paraphrases. We used the S size package of PPDB 2.0 (Pavlick et al., 2015) for high precision. At most 10 candidate paraphrases were considered. We mined paraphrase rules from WikiAnswers (Fader et al., 2014) as described in Section 2.1.3. The extracted rules were ranked using the pointwise mutual information between template pairs in the WikiAnswers corpus. The top 10 candidate paraphrases were used.

Training For the paraphrase scoring model, we used GloVe (Pennington et al., 2014) vectors[3] pretrained on Wikipedia 2014 and Gigaword 5 to initialize the word embedding matrix. We kept this matrix fixed across datasets. Out-of-vocabulary words were replaced with a special unknown symbol. We also augmented questions with start-of- and end-of-sequence symbols. Word vectors for these special symbols were updated during training. Model hyperparameters were validated on the development set. The dimensions of hidden vectors and word embeddings were selected from $\{50, 100, 200\}$ and $\{100, 200\}$, respectively. The dropout rate was selected from $\{0.2, 0.3, 0.4\}$. The BILSTM for the answer sentence selection QA model used the same hyperparameters. Parameters were randomly initialized from a uniform distribution $\mathcal{U}(-0.08, 0.08)$. The learning rate and decay rate of RMSProp were 0.01 and 0.95, respectively. The batch size was set to 150. To alleviate the exploding gradient problem (Pascanu et al., 2013), the gradient norm was clipped to 5. Early stopping was used to determine the number of epochs.

3.3 Paraphrase Statistics

Table 3 presents descriptive statistics on the paraphrases generated by the various systems across datasets (training set). As can be seen, the average paraphrase length is similar to the average length of the original questions. The NMT method generates more paraphrases and has wider coverage, while the average number and coverage of the other two methods varies per dataset. As a way of quantifying the extent to which rewriting takes place, we report BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) scores between the original questions and their paraphrases. The NMT

[2] github.com/sebastien-j/LV_groundhog

[3] nlp.stanford.edu/projects/glove

Metric	GRAPHQ			WEBQ			WIKIQA		
	NMT	PPDB	Rule	NMT	PPDB	Rule	NMT	PPDB	Rule
avg($\|q\|$)		10.87			7.71			6.47	
avg($\|q'\|$)	10.87	12.40	10.51	8.13	8.55	7.54	6.60	7.85	7.15
avg($\#q'$)	13.85	3.02	2.50	13.76	0.71	7.74	13.95	0.62	5.64
Coverage (%)	99.67	73.52	31.16	99.87	35.15	83.61	99.89	31.04	63.12
BLEU (%)	42.33	67.92	54.23	35.14	56.62	42.37	32.40	54.24	40.62
TER (%)	39.18	14.87	38.59	45.38	19.94	43.44	46.10	17.20	48.59

Table 3: Statistics of generated paraphrases across datasets (training set). avg($|q|$): average question length; avg($|q'|$): average paraphrase length; avg($\#q'$): average number of paraphrases; coverage: the proportion of questions that have at least one candidate paraphrase.

method and the rules extracted from WikiAnswers tend to paraphrase more (i.e., have lower BLEU and higher TER scores) compared to PPDB.

3.4 Comparison Systems

We compared our framework to previous work and several ablation models which either do not use paraphrases or paraphrase scoring, or are not jointly trained.

The first baseline only uses the base QA models described in Section 2.3 (SIMPLEGRAPH and BILSTM). The second baseline (AVGPARA) does not take advantage of paraphrase scoring. The paraphrases for a given question are used while the QA model's results are directly averaged to predict the answers. The third baseline (DATAAUGMENT) employs paraphrases for data augmentation during training. Specifically, we use the question, its paraphrases, and the correct answer to automatically generate new training samples.

In the fourth baseline (SEPPARA), the paraphrase scoring model is separately trained on paraphrase classification data, without taking question-answer pairs into account. In the experiments, we used the Quora question paraphrase dataset[4] which contains question pairs and labels indicating whether they constitute paraphrases or not. We removed questions with more than 25 tokens and sub-sampled to balance the dataset. We used 90% of the resulting 275K examples for training, and the remaining for development. The paraphrase score $s(q'|q)$ (Equation (5)) was wrapped by a sigmoid function to predict the probability of a question pair being a paraphrase. A binary cross-entropy loss was used as the objective. The classification accuracy on the dev set was 80.6%.

[4] goo.gl/kMP46n

Method	Average F1 (%)	
	GRAPHQ	WEBQ
SEMPRE (Berant et al., 2013)	10.8	35.7
JACANA (Yao and Van Durme, 2014)	5.1	33.0
PARASEMP (Berant and Liang, 2014)	12.8	39.9
SUBGRAPH (Bordes et al., 2014a)	-	40.4
MCCNN (Dong et al., 2015)	-	40.8
YAO15 (Yao, 2015)	-	44.3
AGENDAIL (Berant and Liang, 2015)	-	49.7
STAGG (Yih et al., 2015)	-	48.4 (52.5)
MCNN (Xu et al., 2016)	-	47.0 (53.3)
TYPERERANK (Yavuz et al., 2016)	-	**51.6** (52.6)
BILAYERED (Narayan et al., 2016)	-	47.2
UDEPLAMBDA (Reddy et al., 2017)	**17.6**	49.5
SIMPLEGRAPH (baseline)	15.9	48.5
AVGPARA	16.1	48.8
SEPPARA	18.4	49.6
DATAAUGMENT	16.3	48.7
PARA4QA	**20.4**	**50.7**
−NMT	18.5	49.5
−PPDB	19.5	50.4
−RULE	19.4	49.1

Table 4: Model performance on GRAPHQUESTIONS and WEBQUESTIONS. Results with additional task-specific resources are shown in parentheses. The base QA model is SIMPLEGRAPH. Best results in each group are shown in **bold**.

Finally, in order to assess the individual contribution of different paraphrasing resources, we compared the PARA4QA model against versions of itself with one paraphrase generator removed (−NMT/−PPDB/−RULE).

3.5 Results

We first discuss the performance of PARA4QA on GRAPHQUESTIONS and WEBQUESTIONS. The first block in Table 4 shows a variety of systems previously described in the literature using average F1 as the evaluation metric (Berant et al., 2013). Among these, PARASEMP, SUBGRAPH, MCCNN, and BILAYERED utilize paraphrasing resources. The second block compares PARA4QA against various related baselines (see Section 3.4). SIMPLEGRAPH results on WEBQUESTIONS and GRAPHQUESTIONS are taken from Reddy et al. (2016) and Reddy et al. (2017), respectively.

Overall, we observe that PARA4QA outperforms baselines which either do not employ paraphrases (SIMPLEGRAPH) or paraphrase scoring (AVGPARA, DATAAUGMENT), or are not jointly trained (SEPPARA). On GRAPHQUESTIONS, our model PARA4QA outperforms the previous state of the art by a wide margin. Ablation experiments with one of the paraphrase generators removed

Method	MAP	MRR
BIGRAMCNN (Yu et al., 2014)	0.6190	0.6281
BIGRAMCNN+CNT (Yu et al., 2014)	0.6520	0.6652
PARAVEC (Le and Mikolov, 2014)	0.5110	0.5160
PARAVEC+CNT (Le and Mikolov, 2014)	0.5976	0.6058
LSTM (Miao et al., 2016)	0.6552	0.6747
LSTM+CNT (Miao et al., 2016)	0.6820	0.6988
NASM (Miao et al., 2016)	0.6705	0.6914
NASM+CNT (Miao et al., 2016)	0.6886	0.7069
KVMEMNET+CNT (Miller et al., 2016)	**0.7069**	**0.7265**
BILSTM (baseline)	0.6456	0.6608
AVGPARA	0.6587	0.6753
SEPPARA	0.6613	0.6765
DATAAUGMENT	0.6578	0.6736
PARA4QA	**0.6759**	**0.6918**
−NMT	0.6528	0.6680
−PPDB	0.6613	0.6767
−RULE	0.6553	0.6756
BILSTM+CNT (baseline)	0.6722	0.6877
PARA4QA+CNT	**0.6978**	**0.7131**

Table 5: Model performance on WIKIQA. +CNT: word matching features introduced in Yang et al. (2015). The base QA model is BILSTM. Best results in each group are shown in **bold**.

Examples	$p_\theta\,(q'\|q)$
(`music.concert_performance.performance_role`)	
what sort of part do queen play in concert	0.0659
what role do queen play in concert	0.0847
what be the role play by the queen in concert	0.0687
what role do queen play during concert	0.0670
what part do queen play in concert	0.0664
which role do queen play in concert concert	0.0652
(`sports.sports_team_roster.team`)	
what team do shaq play 4	0.2687
what team do shaq play for	0.2783
which team do shaq play with	0.0671
which team do shaq play out	0.0655
which team have you play shaq	0.0650
what team have we play shaq	0.0497

Table 6: Questions and their top-five paraphrases with probabilities learned by the model. The Freebase relations used to query the correct answers are shown in brackets. The original question is underlined. Questions with incorrect predictions are in *red*.

show that performance drops most when the NMT paraphrases are not used on GRAPHQUESTIONS, whereas on WEBQUESTIONS removal of the rule-based generator hurts performance most. One reason is that the rule-based method has higher coverage on WEBQUESTIONS than on GRAPHQUESTIONS (see Table 3).

Results on WIKIQA are shown in Table 5. We report MAP and MMR which evaluate the relative ranks of correct answers among the candidate sentences for a question. Again, we observe that PARA4QA outperforms related baselines (see BILSTM, DATAAUGMENT, AVGPARA, and SEPPARA). Ablation experiments show that performance drops most when NMT paraphrases are removed. When word matching features are used (see +CNT in the third block), PARA4QA reaches state of the art performance.

Examples of paraphrases and their probabilities $p_\theta\,(q'|q)$ (see Equation (6)) learned by PARA4QA are shown in Table 6. The two examples are taken from the development set of GRAPHQUESTIONS and WEBQUESTIONS, respectively. We also show the Freebase relations used to query the correct answers. In the first example, the original question cannot yield the correct answer because of the mismatch between the question and the knowledge base. The paraphrase contains "*role*" in place of "*sort of part*", increasing the chance of overlap between the question and the predicate words. The second question contains an informal expression "*play 4*", which confuses the QA model. The paraphrase model generates "*play for*" and predicts a high paraphrase score for it. More generally, we observe that the model tends to give higher probabilities $p_\theta\,(q'|q)$ to paraphrases biased towards delivering appropriate answers.

We also analyzed which structures were mostly paraphrased within a question. We manually inspected 50 (randomly sampled) questions from the development portion of each dataset, and their three top scoring paraphrases (Equation (5)). We grouped the most commonly paraphrased structures into the following categories: a) question words, i.e., wh-words and and "*how*"; b) question focus structures, i.e., cue words or cue phrases for an answer with a specific entity type (Yao and Van Durme, 2014); c) verbs or noun phrases indicating the relation between the question topic entity and the answer; and d) structures requiring aggregation or imposing additional constraints the answer must satisfy (Yih et al., 2015). In the example "*which year did Avatar release in UK*", the question word is "*which*", the question focus is "*year*", the verb is "*release*", and "*in UK*" constrains the answer to a specific location.

Figure 3 shows the degree to which different types of structures are paraphrased. As can be seen, most rewrites affect Relation Verb, especially on WEBQUESTIONS. Question Focus, Relation NP, and Constraint & Aggregation are more

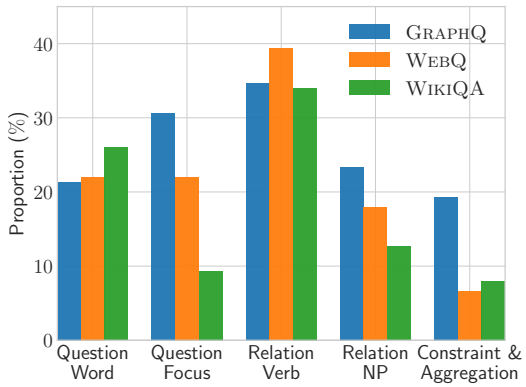

Figure 3: Proportion of linguistic phenomena subject to paraphrasing within a question.

Method	Average F1 (%)	
	Simple	Complex
SIMPLEGRAPH	20.9	12.2
PARA4QA	**27.4** (+6.5)	**16.0** (+3.8)

Table 7: We group GRAPHQUESTIONS into simple and complex questions and report model performance in each split. Best results in each group are shown in **bold**. The values in brackets are absolute improvements of average F1 scores.

often rewritten in GRAPHQUESTIONS compared to the other datasets.

Finally, we examined how our method fares on simple versus complex questions. We performed this analysis on GRAPHQUESTIONS as it contains a larger proportion of complex questions. We consider questions that contain a single relation as simple. Complex questions have multiple relations or require aggregation. Table 7 shows how our model performs in each group. We observe improvements for both types of questions, with the impact on simple questions being more pronounced. This is not entirely surprising as it is easier to generate paraphrases and predict the paraphrase scores for simpler questions.

4 Conclusions

In this work we proposed a general framework for learning paraphrases for question answering. Paraphrase scoring and QA models are trained end-to-end on question-answer pairs, which results in learning paraphrases with a purpose. The framework is not tied to a specific paraphrase generator or QA system. In fact it allows to incorporate several paraphrasing modules, and can serve as a testbed for exploring their coverage and rewriting capabilities. Experimental results on three datasets show that our method improves performance across tasks. There are several directions for future work. The framework can be used for other natural language processing tasks which are sensitive to the variation of input (e.g., textual entailment or summarization). We would also like to explore more advanced paraphrase scoring models (Parikh et al., 2016; Wang and Jiang, 2016) as well as additional paraphrase generators since improvements in the diversity and the quality of paraphrases could also enhance QA performance.

Acknowledgments The authors gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1; Mallinson) and the European Research Council (award number 681760; Lapata).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, Ann Arbor, Michigan. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Rahul Bhagat and Eduard Hovy. 2013. What is a paraphrase? *Computational Linguistics*, 39(3):463–472.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar. Association for Computational Linguistics.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Volume 8724*, ECML PKDD 2014, pages 165–180, New York, NY, USA. Springer-Verlag New York, Inc.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 196–205, Honolulu, Hawaii. Association for Computational Linguistics.

Bo Chen, Le Sun, Xianpei Han, and Bo An. 2016. Sentence rewriting for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 766–777, Berlin, Germany. Association for Computational Linguistics.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 260–269, Beijing, China. Association for Computational Linguistics.

Pablo Duboue and Jennifer Chu-Carroll. 2006. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 33–36, New York City, USA. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618, Sofia, Bulgaria. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1156–1165, New York, NY, USA. ACM.

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1168–1179, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: the paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586, Lisbon, Portugal. Association for Computational Linguistics.

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

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54, Sapporo, Japan.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1188–1196.

Bill MacCartney. 2009. *Natural Language Inference.* Ph.D. thesis, Stanford University.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2016. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, Valencia, Spain.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1727–1736.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of english. *Natural Language Engineering*, 7(3):207–223.

Shashi Narayan, Siva Reddy, and Shay B Cohen. 2016. Paraphrase generation from Latent-Variable PCFGs for semantic parsing. In *Proceedings of the 9th International Natural Language Generation Conference*, pages 153–162, Edinburgh, Scotland.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of*

40th Annual Meeting of the Association for Computational Linguistics, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1310–1318, Atlanta, Georgia.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 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 (Volume 2: Short Papers)*, pages 425–430, Beijing, China. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 285–290.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. *arXiv preprint arXiv:1702.03196*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572, Austin, Texas. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. Technical report.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451, San Diego, California. Association for Computational Linguistics.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2326–2336, Berlin, Germany. Association for Computational Linguistics.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 66–70, Denver, Colorado. Association for Computational Linguistics.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966, Baltimore, Maryland. Association for Computational Linguistics.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 149–159, Austin, Texas. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911, Denver, Colorado. Association for Computational Linguistics.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. In *NIPS Deep Learning Workshop*.