Proceedings of the AAAI 2019 Workshop (WS13) on **Reasoning and Complex Question-Answering** (RCQA-19)

January 28, 2019
Hilton Hawaiian Village, Honolulu
ibm.biz/complexQA

Organizers

Kartik Talamadupula, IBM Research AI
Peter Clark, Allen Institute for Artificial Intelligence (AI2)
Rajarshi Das, Univ of Massachusetts Amherst
Pavan Kapanipathi, IBM Research AI
Mausam, Indian Institute of Technology (IIT) Delhi
Michael Witbrock, IBM Research AI

Understanding Complex Multi-sentence Entity seeking Questions

Danish Contractor[1,2*], **Barun Patra**[1†], **Mausam**[1], **Parag Singla**[1]
[1]Indian Institute of Technology, New Delhi [2]IBM Research AI, New Delhi
dcontrac@in.ibm.com,barunpatra95@gmail.com, mausam,parags@cse.iitd.ac.in

Abstract

We present the novel task of understanding multi-sentence *entity-seeking* questions (MSEQs) i.e, questions that may be expressed in multiple sentences, and that expect one or more entities as an answer. We formulate the problem of understanding MSEQs as a semantic labeling task over an open representation that makes minimal assumptions about schema or ontology specific semantic vocabulary. At the core of our model, we use a BiDiLSTM (bi-directional LSTM) CRF and to overcome the challenges of operating with low training data, we supplement it by using hand-designed features, as well as hard and soft constraints spanning multiple sentences. We find that this results in a 6-7pt gain over a vanilla BiDiLSTM CRF. We demonstrate the strengths of our work using the novel task of answering real-world entity-seeking questions from the tourism domain. The use of our labels helps answer 53% more questions with 42 % more accuracy as compared to baselines.

Introduction

We introduce the novel task of understanding multi-sentence questions. Specifically, we focus our attention on multi-sentence *entity-seeking* questions (MSEQs) i.e., questions that expect one or more entities as answer. Such questions are commonly found in online forums, blog posts, discussion boards etc and come from a variety of domains including tourism, books and consumer products.

Figure 1 shows an example MSEQ from a tourism forum, where the user is interested in finding a hotel that satisfies some constraints and preferences; an *answer* to this question is thus the name of a hotel (entity) which needs to satisfy some properties such as being a 'budget' option. A preliminary analysis of such entity-seeking questions from online forums reveals that almost all of them contain multiple sentences – they often elaborate on a user's specific situation before asking the actual question.

In order to *understand* and answer such a user question, we convert the question into a machine representation consisting of labels identifying the *informative* portions in a question. We are motivated by our work's applicability to a wide variety of domains and therefore choose not to restrict the representation to use a domain-specific vocabulary. Instead, we design an *open* semantic representation, inspired in part by Open QA (Fader, Zettlemoyer, and Etzioni 2014), in which we explicitly annotate the answer (entity) type; other answer attributes, while identified, are not further categorized. Eg. in Figure 1 'place to stay' is labeled as $entity.type$ while 'budget' is labeled as an $entity.attr$. We also allow attributes of the *user* to be represented. Domain specific annotations such as $location$ for tourism questions are permitted. Such labels are then be supplied to a downstream information retrieval (IR) or a QA component to directly present an answer entity.

We pose the task of understanding MSEQs as a semantic labeling (shallow parsing)[1] task where tokens from the question are annotated with a semantic label from our open representation. However, in contrast to related literature on semantic role labeling (Yang and Mitchell 2017), slot filling tasks (Bapna et al. 2017) and query formulation (Wang and Nyberg 2016; Vtyurina and Clarke 2016; Nogueira and Cho 2017), semantic parsing of MSEQs raise several novel challenges. MSEQs express a wide variety of intents and requirements which span across multiple sentences, requiring the model to capture within-sentence as well as inter-sentence interactions effectively. In addition, questions can be unnecessarily belabored requiring the system to reason about what is important and what is not. Lastly, we find that generating training data for parsing MSEQs is hard due to the complex nature of the task, further requiring the models to operate in low training data settings.

In order to address these challenges and label MSEQs, we use a bi-directional LSTM CRF (BiDiLSTM CRF) (Huang, Xu, and Yu 2015) and extend it in two ways: (i) We encode knowledge by incorporating hand-designed features as well as semantic constraints over the entire multi-sentence question during end-to-end training. (ii) We use partially labeled questions, that are easier to source, to improve training.

In summary, our paper makes the following contributions:

1. We present the novel task of understanding multi-sentence entity-seeking questions (MSEQs). We define *open* se-

*This work was carried out as part of PhD research at IIT Delhi. The author is also a regular employee at IBM Research AI.

†Work carried out when Barun was an undergraduate student at IIT Delhi.

[1]We use the phrases 'semantic labeling' and 'semantic parsing' interchangeably in this paper.

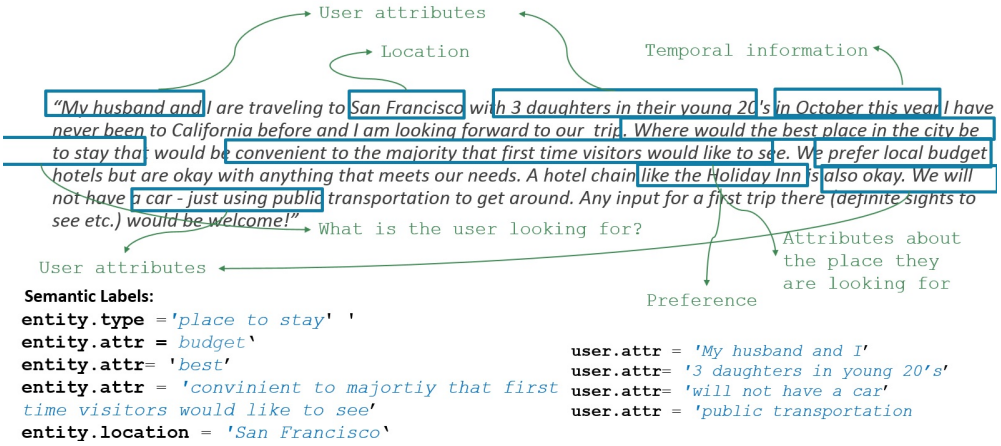


Figure 1: An entity-seeking MSEQ and annotated with our semantic labels

mantic labels, which minimize schema or ontology specific semantic vocabulary and can easily generalize across domains. These semantic labels identify *informative* portions of a question that can be used by a downstream answering component.

2. The core of our model uses a BiDiLSTM CRF model. We extend this by providing hand-designed features and using Constrained Conditional Model (CCM) (Chang, Ratinov, and Roth 2007) inference, which allows us to specify within-sentence as well as inter-sentence (hard and soft) constraints, encoding prior knowledge about the labeling task.
3. We present detailed experiments on our models using the tourism domain as an example. We also demonstrate how crowd-sourced partially labeled questions can be effectively used in our constraint based tagging framework to help improve labeling accuracy. We find that our best model achieves 7 pt improvement in F1 scores over a baseline BiDiLSTM CRF.
4. We demonstrate the applicability of our semantic labels in an end-QA task: the novel task of directly answering tourism-related MSEQs using a web based semi-structured knowledge source. Our semantic labels help formulate a more effective query to knowledge sources and our system answers 53% more questions with 42 % more accuracy as compared to baselines

Related Work

To the best of our knowledge, we are the first to explicitly address the task of *understanding* multi-sentence entity-seeking questions and demonstrate its use in an answering task.

Question Answering Systems

There are two common approaches for QA systems – joint and pipelined, both with different advantages. The joint systems usually train an end-to-end neural architecture, with a softmax over candidate answers (or spans over a given passage) as the final layer (Iyyer et al. 2014; Rajpurkar et al. 2016), while a pipelined approach (Fader, Zettlemoyer, and Etzioni 2014; Berant and Liang 2014; Fader, Zettlemoyer, and Etzioni 2013; Kwiatkowski et al. 2013; Vtyurina and Clarke 2016; Wang and Nyberg 2016) divides the task into two components – question processing (understanding) and querying the knowledge source. Our work follows the second approach.

In this paper, we return entity answers to multi-sentence entity seeking questions. The problem of returning direct, (non-document/passage) answers to questions from background knowledge sources has been studied, but primarily for single sentence factoid-like questions (Fader, Zettlemoyer, and Etzioni 2014; Berant and Liang 2014; Yin et al. 2015; Sun et al. 2015; Saha et al. 2016; Khot, Sabharwal, and Clark 2017; Lukovnikov et al. 2017). Reading comprehension tasks (Rajpurkar et al. 2016; Trischler et al. 2016; Joshi et al. 2017; Trivedi et al. 2017) require answers to be generated from unstructured text also only return answers for simple single-sentence questions.

Other works have considered multi-sentence questions, but in different settings, such as the specialized setting of answering multiple-choice SAT and science questions (Seo et al. 2015; Clark et al. 2016; Khot, Sabharwal, and Clark 2017; Guo et al. 2017), mathematical word problems (Liang et al. 2016), and textbook questions (Sachan, Dubey, and Xing 2016). Such systems do not return entity answers to questions. Community QA systems (Bogdanova and Foster 2016; Shen et al. 2015; Qiu and Huang 2015; Tan, Xiang, and Zhou 2015) match questions with *user*-provided answers, instead of entities from background knowledge-source. IR-based systems (Vtyurina and Clarke 2016; Wang and Nyberg 2016) query the Web for open-domain questions, but return long (1000 character) passages as answers; they haven't been developed for, or tested on entity-seeking questions. These techniques that can handle MSEQs (Vtyurina and Clarke 2016; Wang and Nyberg 2016) typically perform retrieval using keywords extracted from questions; these do not "understand" the questions and can't answer many tourism questions, as our experiments

show. The more traditional solutions (e.g., semantic parsing) that parse the questions deeply can process only *single*-sentence questions (Fader, Zettlemoyer, and Etzioni 2014; Berant and Liang 2014; Fader, Zettlemoyer, and Etzioni 2013; Kwiatkowski et al. 2013).

Finally, systems such as QANTA (Iyyer et al. 2014) also answer complex multi-sentence questions but their methods can only select answers from a small list of entities and also require large amounts of training data with redundancy of QA pairs. In contrast, the subset of Google Places we experiment with has close to half a million entities.

We discuss literature on parsing (understanding) questions in the next section.

Question Parsing

QA systems use a variety of different intermediate semantic representations. Most of them, including the rich body of work in NLIDB and semantic parsing, parse *single* sentence questions into a query based on the underlying ontology or DB schema and are often learned directly by defining grammars, rules and templates (Zettlemoyer 2009; Liang 2011; Kwiatkowski et al. 2013; Berant et al. 2013; Yih et al. 2015; Sun et al. 2015; Saha et al. 2016; Reddy et al. 2016; Khot, Sabharwal, and Clark 2017; Cheng et al. 2017; Lukovnikov et al. 2017). Work such as (Fader, Zettlemoyer, and Etzioni 2014; Berant and Liang 2014) build *open* semantic representations for single sentence questions, that are not tied to a specific knowledge source or ontology. We follow a similar approach and develop an open semantic representation for multi-sentence entity seeking questions. Our representation uses labels that help a downstream answering component return entity answers.

Recent works build neural models that represent a question as a continuous-valued vector (Bordes, Chopra, and Weston 2014; Bordes, Weston, and Usunier 2014; Xu et al. 2016; Chen et al. 2016; Zhang et al. 2016) but such methods require significant amounts of training data. Some systems rely on IR and do not construct explicit semantic representations at all (Sun et al. 2015; Vtyurina and Clarke 2016); they rely on selecting keywords from the question for querying and as shown in our experiments do not perform well for answering multi-sentence entity-seeking questions. Work such as that by (Nogueira and Cho 2017) uses reinforcement learning to select query terms in a document retrieval task and requires a large collection of document-relevance judgments. Extending such an approach for our task could be an interesting extension for future work.

We now summarize recent methods employed to generate semantic representations of questions.

Neural Semantic Parsing

There is a large body of literature dealing with semantic parsing of single sentences, especially for frames in PropBank and Framenet (Palmer, Gildea, and Kingsbury 2005; Baker, Fillmore, and Lowe 1998). Most recently, methods that use neural architectures for SRL (Semantic Role Labeling) have been developed. For instance, work by (Zhou and Xu 2015) uses a BiDiLSTM CRF for labeling sentences with PropBank predicate argument structures, while work by (He et al. 2017; 2018) relies on a BiDiLSTM with BIO-encoding constraints during LSTM decoding. Other recent work by tomemnlp2017 proposes a BiDiLSTM CRF model that is further used in a graphical model that encodes SRL structural constraints as factors. Work such as that by (Bapna et al. 2017) uses a BiDiLSTM tagger for predicting task oriented information slots from sentences. Our work uses similar approaches for parsing MSEQs, but we note that such systems cannot be directly used in our task due to their model specific optimizations for their label space. However, we adapt the label space of the recent Deep SRL system (He et al. 2017) for our task and use its predicate tagger as a baseline for evaluation.

Problem Statement

Given a multi-sentence entity seeking question, our goal is to first parse and generate a semantic representation of the question. These semantic labels identify *informative* portions of a question that can be used by a downstream answering component. The semantic representation of the question is then used to return an entity answer for the question using a knowledge source.

Semantic Labels for MSEQs

As mentioned earlier, our question understanding component parses an MSEQ into an *open* semantic representation. Our choice of representation is motivated by two goals. First, we wish to make minimal assumptions about the domain of the QA task and therefore, minimize domain-specific semantic vocabulary[2]. Second, we wish to identify only the *informative* elements of a question, so that a robust downstream QA or IR system can meaningfully answer it. As a first step towards a generic representation for an MSEQ, we make the assumptions that a multi-sentence question is asking only one final question, and that the expected answer is one or more entities. This precludes Boolean, comparison, 'why'/'how', and multiple part questions

We have two labels associated with the entity being sought: *entity.type* and *entity.attr*, to capture the type and the attributes of the entity, respectively. We also include a label *user.attr* to capture the properties of the user asking the question. The semantic labels of $entity.type$ and $entity.attr$ are generic and will be applicable to any domain. Other generic labels to identify related entities (eg: in questions where users ask for entities similar to a list of entities) could also be defined. We also allow the possibility of incorporating additional labels which are domain specific. For instance, for the tourism domain, location could be important, so we can include an additional label $entity.location$ describing the location of the answer entity.

Figure 1 illustrates the choice of our labels with an example from the tourism domain. Here, the user is interested in finding a 'place to stay' ($entity.type$) that satisfies some properties such as 'budget' ($entity.attr$). The question includes some information about the user herself e.g., 'will not have a car' which may become relevant for answering

[2]Our representation can easily be generalized to include domain-specific semantic labels, if required.

the question. The phrase 'San Francisco' describes the location of the entity and is labeled with a domain specific label ($entity.location$).

MSEQ Semantic Parsing

We formulate the task of outputting the semantic representation for a user question as a sequence labeling problem. There is a one to one correspondence between our token-level label set and the semantic labels described in earlier. We utilize a BiDiLSTM CRF (Huang, Xu, and Yu 2015) for sequence labeling and as described previously, we extend the model in order to address the challenges posed by MSEQs: (a) First, we incorporate hand-engineered features especially designed for our labeling task (b) Second, we make use of a Constrained Conditional Model (CCM) (Chang, Ratinov, and Roth 2007) to incorporate within-sentence as well as inter-sentence constraints. These constraints act as a prior and help ameliorate the problems posed by our low-data setting. (c) Third, we use Amazon Mechanical Turk (AMT) to obtain additional partially labeled data which we use in our constraint driven framework.

Features

We incorporate a number of (domain-independent) features into our BiDiLSTM CRF model where each unique feature is represented as a one-hot vector and concatenated with the word-vector representation of each token.

Our features are described as follows: (a) Lexical features for capitalization, indicating numerals etc., token-level features based on POS and NER (b) hand-designed $entity.type$ and $entity.attr$ specific features. These include indicators for guessing potential types, based on targets of WH (*what*, *where*, *which*) words and certain verb classes; multi-sentence features that are based on dependency parses of individual sentences that aid in attribute detection, e.g., for every noun and adjective, an attribute indicator feature is on if any of its ancestors is a potential $type$ as indicated by type feature; indicator features for descriptive phrases (Contractor, Mausam, and Singla 2016), such as adjective-noun pairs. (c) For each token, we include cluster ids generated from a clustering of word2vec vectors (Mikolov et al. 2013) run over a large tourism corpus. (d) We also use the counts of a token in the entire post, as a feature for that token (Vtyurina and Clarke 2016).

Constraints

Since we label multiple-sentence questions, we need to capture patterns spanning across sentences. One alternative would be to model these patterns as features defined over non-adjacent tokens (labels). But this can make the modeling quite complex. Instead, we model them as global constraints over the set of possible labels.

We design the following constraints: (i) type constraint (hard): every question must have at least one $entity.type$ token, and (ii) attribute constraint (soft), which penalizes absence of an $entity.attr$ label in the sequence. (iii) a soft constraint that prefers all $entity.type$ tokens occur in the same sentence. The last constraint helps reduce erroneous $entity.type$ labels but allows the labeler to choose $entity.type$-labeled tokens from multiple sentences only if it is very confident. Thus, while the first two constraints are directed towards improving recall, the last constraint helps improve precision of $entity.type$ labels

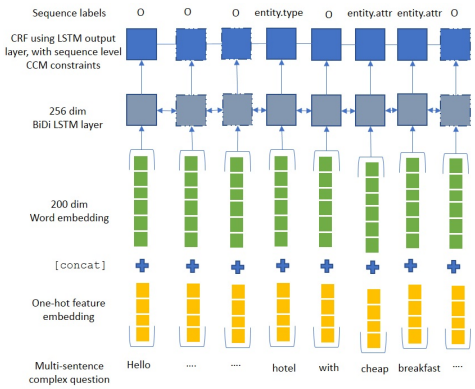


Figure 2: BiDi LSTM CCM for sequence labeling.

In order to use our constraints, we employ Constrained Conditional Models (CCMs) for our task (Chang, Ratinov, and Roth 2007) which use an alternate learning objective expressed as the difference between the original log-likelihood and a constraint violation penalty:

$$\sum_i w^T \phi(\mathbf{x^{(i)}}, \mathbf{y^{(i)}}) - \sum_i \sum_k \rho_k d_{C_k}(\mathbf{x^{(i)}}, \mathbf{y^{(i)}}) \quad (1)$$

Here, i indexes over all examples and k over all constraints. $\mathbf{x^{(i)}}$ is the i^{th} sequence and $\mathbf{y^{(i)}}$ its labeling. ϕ and w are feature and weight vectors respectively. d_{C_k} and ρ_k denote the violation score and weight associated with k^{th} constraint. The w parameters are learned analogous to a vanilla CRF and computing ρ parameters resorts to counting. Hard constraints have an infinite weight. Inference in CCMs is formulated as an Integer Linear Program (ILP); see Chang et al.(2007) for details. The original CCM formulation was in the context of regular CRFs (Lafferty, McCallum, and Pereira 2001) and and we extend its use in a combined model of BiDiLSTM CRF with CCM constraints that is trained end-to-end (Figure 2).

Partially Labeled Data

Data Collection: In order to obtain a larger amount of labeled data for our task, we make use of crowd-sourcing (Amazon Mechanical Turk). Since our labeling task can be fairly complex, we divide our crowd task into multiple steps. We first ask crowd to (i) filter out forum questions that are not entity-seeking questions. For the questions that remain, the crowd provides (ii) $user.*$ labels, and (iii) $entity.*$ labels. Taking inspiration from (He, Lewis, and Zettlemoyer 2015), for each step, instead of directly asking for token labels, we ask a series of indirect questions as described in the next section that can help source high precision annotations.

We obtain two sets of labels (different workers) on each question. However, due to the complex nature of the task we

	type	*attr*	*loc*
Avg. token level agreement	47.98	37.78	68.56

Table 1: Agreement for $entity$ labels on AMT

find that workers are not complete in their labeling and we therefore only use token labels where both the set of workers agreed on labels. Thus we are able to source annotations with high precision, while recall can be low. Table 1 shows token-level agreement statistics for labels collected over a set of 400 MSEQs from the tourism domain. Some of the disagreement arises from labeling errors due to complex nature of the task. In other cases, the disagreement results from their choosing one of the several possible correct answers. E.g., in the phrase "*good restaurant for dinner*" one worker labels $entity.type$ ='restaurant', $entity.attr$ ='good' and $entity.attr$ ='dinner', while another worker simply chooses the entire phrase as $entity.type$.

Training with partially labeled posts We devise a novel method to use this partially labeled data, along with our small training set of expert labeled data, to learn the parameters of our CCM model. We utilize a modified version of Constraints driven learning (CODL) (Chang, Ratinov, and Roth 2007) which uses a semi-supervised iterative weight update algorithm, where the weights at each step are computed using a combination of the models learned on the labeled and the unlabeled set (Chang, Ratinov, and Roth 2007).

Given a dataset consisting of a few fully labeled as well as unlabeled examples, the CoDL learning algorithm first learns a model using only the labeled subset. This model is then used to find labels (in a hard manner) for the unlabeled examples while taking care of constraints. A new model is then learned on this newly annotated set and is combined with the model learned on the labeled set in a linear manner. The parameter update can be described as:

$$(w^{(t+1)}, \rho^{(t+1)}) = \gamma(w^{(0)}, \rho^{(0)}) + (1-\gamma)\mathrm{Learn}(\mathrm{U}^{(t)}) \quad (2)$$

Here, t denotes the iteration number, $U^{(t)}$ denotes the unlabeled examples and Learn is a function that learns the parameters of the model. In our setting, Learn trains the neural network via back-propagation. Instead of using unlabeled examples in $U^{(t)}$ we utilize the partially labeled set whose values have been filled in using parameters at iteration t and, inference over the set involves predicting only the missing labels. This is done using the ILP based formulation described previously and γ controls the relative importance of the labeled and partial examples. To the best our knowledge, we are the first to exploit partial supervision from a crowd-sourcing platform in this manner.

Experimental Evaluation

The goal of our experimental evaluation was to analyze the effectiveness of our proposed model for the task of understanding MSEQs. We next describe our dataset, evaluation methodology and our results in detail.

Dataset

For our current evaluation, we used the following three semantic labels: $entity.type$, $entity.attr$, $entity.location$. We also used a default label $other$ to mark any tokens not matching any of the semantic labels.

We use 150 expert-annotated tourism forum questions (9200 annotated tokens) as our labeled dataset and perform leave-one out cross-validation. This set was labeled by two experts, including one of the authors, with high agreement. For experiments with partially labeled learning, we add 400 partially-annotated questions from crowd-sourced workers to our training set. As described previously, each question is annotated by two workers and we retain token labels marked the same by two workers, while treating the other labels as unknown. We still compute a leave one out cross-validation on our original 150 expert-annotated questions (complete crowd data is included in each training fold).

Methodology

Sequence-tagged tokens identify *phrases* for each semantic label – so, instead of reporting metrics at the token level, we compute a more meaningful joint metric over tagged phrases. We define a matching-based metric that first matches each extracted segment with the closest one in the gold set, and then computes segment level precision using constituent tokens. Analogously, recall is computed by matching each segment in gold set with the best one in extracted set. As an example, for Figure 1, if the system extracts "convenient to the majority" and "local budget" for $entity.attr$ then our matching-metric will compute precision as 0.75 (1.0 for "convenient to the majority" and 0.5 for "local budget)" and recall as 0.45 (1.0 for "budget", 0.0 for "best" and 0.364 for "convenient to the majority ... like to see").

We use the Mallet toolkit[3] for CRF implementation and the GLPK ILP-based solver[4] for CCM inference. In the case of BiDiLSTM based CRF, the BiDiLSTM network at each time step feeds into a linear chain CRF layer. The input states in the LSTM are modeled using a 200 dimension word vector representation of the token. These word vector representations were with pre-trained using the word2vec model(Mikolov et al. 2013) on a large collection of $80,000$ tourism questions. For CoDL learning we set γ to 0.9 as per original authors' recommendations.

Results

Table 2 reports the performance of our semantic labeler under different configurations. We find that a BiDiLSTM CRF (lower half of the table) approach outperforms a CRF system (upper half of the table) in each comparable setting - for instance, using a baseline vanilla CRF based system using all features gives us an aggregate F1 of 50.8 while the the performance of a BiDi LSTM CRF approach using features is 56.2. As a baseline we use the predicate tagger from the Deep SRL system (He et al. 2017) to utilize our label space and we find that it performs similar to our CRF setup. Our

[3] http://mallet.cs.umass.edu/

[4] https://www.gnu.org/software/glpk/

Model	F1 (entity.type)	F1 (entity.attr)	F1 (entity.loc)	F1 (aggr)
Deep SRL (He et al. 2017)	48.4	47.8	53.2	49.8
CRF (all features)	51.4	45.3	55.7	50.8
CCM	59.6	50.0	56.1	55.2
CCM (with all crowd data)	55.1	42.2	46.7	48.0
PS CCM	58.5	**50.6**	60.3	56.5
BiDi LSTM CRF	53.3	47.6	52.1	51.0
BiDi LSTM CRF+Feat	58.4	48.1	62.0	56.2
BiDi LSTM CCM+Feat	59.4	49.8	**62.3**	57.2
PS BiDi LSTM CCM+Feat	**62.9**	50.4	61.5	**58.3**

Table 2: Sequence tagger $F1$ scores using CRF with all features (feat), CCM with all features & constraints, and partially-supervised CCM over partially labeled crowd data. The second set of results mirror these settings using a bi-directional LSTM CRF. Results are statistically significant (paired t-test,p value$<$0.000124 for aggregate $F1$). Models with "PS" as a prefix use partial supervision.

Algorithm	Prec	Recall	F1
CRF (all features)	66.9	41.7	51.4
CCM (all features)	62.1	57.2	59.6
BiDI LSTM CRF with Features	54.1	63.6	58.4
BiDi LSTM CCM with Features	55.1	64.5	59.4

Table 3: (i) Precision and Recall of *entity.type* with and without CCM inference.

best model combines a BiDiLSTM CRF with hand-designed features, CCM constraints along with learning from partially annotated crowd data. This model has nearly a 7 pt gain over the baseline BiDiLSTM CRF model. Further, usage of hand-curated features, within-sentence and cross-sentence constraints as well as partial supervision, each help successively improve the results. Next, we study the effect of each of these enhancements in detail.

Effect of features In an ablation study performed to learn the incremental importance of each feature, we find that descriptive phrases, and our hand-constructed multi-sentence type and attribute indicators improve the performance of each label by 2-3 points. Word2vec features help type detection because $entity.type$ labels often occur in similar contexts, leading to informative vectors for typical type words. Frequency of non stopword words in the multi-sentence post are an indicator of the word's relative importance, and the feature also helps improves overall performance.

Effect of constraints A closer inspection of Table 2 reveals that the vanilla CRF configuration sees more benefit in using our CCM constraints as compared to the BiDiLSTM CRF based setup (4pt vs 1pt). To understand why, we study the detailed precision-recall characteristics of individual labels; the results for $entity.type$ are reported in Table 3. We find that the BiDiLSTM CRF based setup has significantly higher recall than its equivalent vanilla CRF counterpart while the opposite trend is observed for precision. As a result, since two of the three constraints employed by us in CCM are oriented towards improving recall[5], we find that they improve overall F1 more by finding tags that were otherwise of lower probability (i.e. improving recall).

Effect of partial-supervision In order to further understand the effect of partial-supervision, we trained a CCM based model that makes use of *all* the crowd-sourced labels for training, by adding conflicting labels for a question as two independent training data points. As can be seen, using the entire noisy crowd-labeled sequences (row labeled "CCM (with all crowd data)" in upper half of Table 2) hurts the performance significantly resulting in an aggregate $F1$ of just 48.0 while the corresponding partially-supervised CCM system trained using partially labeled data has an $F1$ of 56.5.

Overall: Our results demonstrate that the use of each of hand-engineering features, within-sentence and inter-sentence constraints and use of partially labeled data help improve the accuracy of labeling MSEQs.

MSEQ-QA

We now demonstrate the usefulness of our MSEQ semantic labels and tagging framework by enabling a QA end task which returns entity answers for multi-sentence MSEQs. To the best of our knowledge we are the first to attempt such a QA task.

We use our sequence tagger described previously to generate the semantic labels of the questions. These semantic labels and their targets are used to formulate a query to the Google Places collection, which serves as our knowledge source.[6]. The Google places collection contains details about eateries, attractions, hotels and other points of interests from all over the world, along with reviews and ratings from users. It exposes an end point that can be used to execute free text queries and it returns entities as results.

We convert the semantic-labels tagged phrases into a Google Places query via the transformation: "concat($entity.attr$) $entity.type$ in $entity.location$". Here, concat lists all attributes in a space-separated fashion. Since some of the attributes may be negated in the original question, we filter out these attributes and do not include it as part of the query for Google Places.

Detection of Negations: We use a list of *triggers* that indicate negation. We start with a manually curated set of seed words, and expand it using synonym and antonym counter fitted word vectors (Mrksic et al. 2016). The resulting set of *trigger* words flag the presence of a negation in a sentence. We also define the scope of a negation trigger as a token (or a set of continuous tokens with the same label) labeled by our sequence tagger that occur within a specified window of the trigger word. Table reports the accuracy of our negation rules as evaluated by an author. The 'Gold' columns denote the performance when using gold semantic label mentions. The 'System' columns are the performance when using labels generated by our sequence tagger.

[5]Recall that we require at least one *entity.type* (hard constraint) and prefer at least one *entity.attr* (soft constraint)

[6]*https://developers.google.com/places/web-service/*

	Gold			System		
	P	R	F1	P	R	F1
Negations	86	66	74.6	85	62	71.7

Table 4: Performance of negation detection using gold sequence labels, and system generated labels

System	Acc@3 (%)	MRR	Recall (%)
WebQA	18.8	0.16	40
WebQA (manual)	40.2	0.37	31.2
MSEQ-QA	**56.9**	**0.47**	**47.8**

Table 5: QA task results using the Google Places web API as knowledge source.

Baseline Since there are no baselines for this task, we adapt and re-implement a recent complex QA system (called WebQA) originally meant for finding appropriate Google results (documents) to questions posed in user forums (Vtyurina and Clarke 2016). WebQA first short-lists a set of top 10 words in the question using a tf-idf based scheme computed over the set of all questions. A supervised method is then used to further, shortlist 3-4 words from that form the query. For best performance, instead of using supervised learning for further shortlisting keywords (as in the original paper), in our implementation an expert chooses 3-4 best words manually. This query on executed against the Google places collection returns answer entities instead of documents.

We randomly select 300 new unseen questions (different from the questions used in the previous section), from a tourism forum website and manually remove 105 of those that were not entity-seeking. The remaining 195 questions forms our test set. Our annotators manually check each entity-answer returned by the systems for correctness. Inter-annotator agreement for relevance of answers measured on 1300+ entities from 100 questions was 0.79. Evaluating whether an entity answer returned is correct is subjective and time consuming. For each entity answer returned, annotators need to manually query a web-search engine to evaluate whether an entity returned by the system adequately matches the requirements of the user posting the question. Given the subjective and time consuming nature of this task, we believe 0.79 is an adequate level of agreement on entity answers.

MSEQ-QA: Results Results: Table 5 reports Accuracy@3, which gives credit if any one of the top three answers is a correct answer. We also report Mean Reciprocal Rank (MRR). Both of these measures are computed only on the subset of attempted questions (any answer returned). Recall is computed as the percentage of questions answered correctly within the top three answers over all questions. In case the user question requires more than one entity type[7], we mark an answer correct as long as one of them is attempted and answered correctly. Note that these answers are ranked by Google Places based on relevance to the query. As can be seen, the use of our semantic labels (MSEQ-QA) results in nearly 17 point higher accuracy with a 16 point higher recall compared to WebQA (manual), because of a more directed & effective query to Google Places collection.

Overall, our semantic labels based QA system (MSEQ-QA) answers approximately 48% of the questions with an accuracy of 57% for this challenging task of answering MSEQs.

MSEQ-QA : Qualitative Study and Error Analysis Table 6 presents some examples of questions answered by the MSEQL based QA system. As can be seen our system supports a variety of question intents/entities and due to our choice of an open semantic representation, we are not limited to specific entity types, entity instances, attributes or locations. For example, in $Q1$ the user is looking for "local dinner suggestions" on Christmas eve, and the answer entity returned by our system is to dine at the "St. Peter Stiftskulinarium" in Salzburg, while in $Q2$ the user is looking for recommendations for "SOM tours" (Sound of Music Tours). $Q3$ is incorrect because the entity returned does not fulfill the location constraints of being close to the "bazar" while $Q5$ returns an incorrect entity type. $Q4$ is a complicated question with strict location, budget & attribute constraints.
Error Analysis: We conducted a detailed error study on 105 of the test set questions and we find that approximately 58% of questions were not answered by our system due to limitations of the knowledge source while approximately 42% of the recall loss in the system can be traced to errors in the semantic labels.

Conclusion and Future Work

We have presented the novel task of understanding entity-seeking multi-sentence questions. MSEQs are an important class of questions, as they appear frequently on online forums. They expose novel challenges for semantic parsing as they contain multiple sentences requiring cross-sentence interactions and also need to be built in low data settings due to challenges associated with sourcing training data. We define a set of open semantic labels that we use to formulate a multi-sentence question parsing task.

Our solution consists of sequence labeling based on a BiDiLSTM CRF model. We use hand-engineered features, inter-sentence CCM constraints, and partially-supervised training, enabling the use of crowdsourced incomplete annotation. We find these methods results in a 7pt gain over baseline BiDiLSTM CRFs. We further demonstrate the strength of our work by applying the semantic labels towards a novel end-QA task that returns entity answers for MSEQs from a web API based unstructured knowledge source that outperforms baselines.

An error analysis on our test set suggests the need for a deeper IR system that parses constructs from our semantic representation to execute multiple sub-queries. As another direction of work, we would like to train an end to end neural system to solve our QA task. This would require generating a large dataset of labeled QA pairs which could perhaps be sourced semi-automatically using data available in tourism QA forums.

[7] A question can ask for multiple things, eg., 'museums' as well suggestions for "hotels".

No.	Question	Entity Type	System Answer
1	My family and my brother's family will be in Salzburg over Christmas 2015. We have arranged to do the Sleigh Ride on Christmas day but are keen to do a local style Christmas Day dinner somewhere. Any suggestions?	Special Dinner place	St. Peter Stiftskulinarium, Sankt-Peter-Bezirk 14, 5020 Salzburg
2	Heading to Salzburg by car on Friday September 18th with my wife and her parents (70's) and trying to make the most of the one day. Thinking about a SOM tour, but not sure what the best tour is, not a big fan of huge groups or buses, but would sacrifice for my Mother in Law (LOL). Also thinking about Old Town or the Salzburg Fortress. Any suggestions? Where to park to have easy access as well as a great place for dinner.Thanks so much!	Tour	Bob's Special Tours, Rudolfskai 38, 5020 Salzburg, Austria
3	I am planning to visit Agra for 2 days in mid Dec with my friends.My plan is to try some street food and do some local shopping on day 1 and thus wish to stay in a good budget 3 star hotel (as I wont be spending much time in the hotel) at walking distance from such street foodlocal shopping market.Then on the 2nd day, I want to just relax and enjoy the hotel.(I have booked a premium category hotel, Radisson Blu for this day hoping for a relaxed stay)Please suggest some good hotel or market around which I should book an hotel for my first day.	Hotel with location constraints	Hotel Taj Plaza, Agra, Taj Mahal East Gate, Near Hotel Oberoi Amar Vilas, VIP Road, Shilpgram, Agra, Uttar Pradesh 282001, India
4.	Hi,I am looking for a good hotel in Shillong (preferably near Police bazar) which would offer free wifi, spa and are okay with unmarried couples. My budget is 3k maximum. please suggest the best place to stay.	Hotel with location and budget constraints	Hotel Pegasus Crown, Ward's Lake Road, Police Bazar, Shillong, Meghalaya 793001, India ;

Table 6: Some sample questions from our test set and the answers returned by our system. Answers in green are identified as correct while those in red are incorrect.

References

Baker, C. F.; Fillmore, C. J.; and Lowe, J. B. 1998. The berkeley framenet project. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*, COLING '98, 86–90. Stroudsburg, PA, USA: Association for Computational Linguistics.

Bapna, A.; Tur, G.; Hakkani-Tur, D.; and Heck, L. 2017. Towards zero shot frame semantic parsing for domain scaling. In *Interspeech 2017*.

Berant, J., and Liang, P. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1533–1544.

Bogdanova, D., and Foster, J. 2016. This is how we do it: Answer reranking for open-domain how questions with paragraph vectors and minimal feature engineering. 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*, 1290–1295.

Bordes, A.; Chopra, S.; and Weston, J. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 615–620.

Bordes, A.; Weston, J.; and Usunier, N. 2014. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I*, 165–180.

Chang, M.; Ratinov, L.; and Roth, D. 2007. Guiding semi-supervision with constraint-driven learning. In *In Proc. of the Annual Meeting of the ACL*.

Chen, L.; Jose, J. M.; Yu, H.; Yuan, F.; and Zhang, D. 2016. A semantic graph based topic model for question retrieval in community question answering. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, 287–296. New York, NY, USA: ACM.

Cheng, J.; Reddy, S.; Saraswat, V.; and Lapata, M. 2017. Learning structured natural language representations for semantic parsing. *arXiv preprint arXiv:1704.08387*.

Clark, P.; Etzioni, O.; Khot, T.; Sabharwal, A.; Tafjord, O.; Turney, P.; and Khashabi, D. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 2580–2586. AAAI Press.

Contractor, D.; Mausam; and Singla, P. 2016. Entity-balanced gaussian plsa for automated comparison. In *Proceedings of NAACL-HLT*, 69–79.

Fader, A.; Zettlemoyer, L. S.; and Etzioni, O. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, 1608–1618.

Fader, A.; Zettlemoyer, L.; and Etzioni, O. 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, 1156–1165. New York, NY, USA: ACM.

Guo, S.; Liu, K.; He, S.; Liu, C.; Zhao, J.; and Wei, Z. 2017. Ijcnlp-2017 task 5: Multi-choice question answering in examinations. In *IJCNLP*, 34–40.

He, L.; Lee, K.; Lewis, M.; and Zettlemoyer, L. 2017. Deep

semantic role labeling: What works and what's next. 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*, 473–483.

He, L.; Lee, K.; Levy, O.; and Zettlemoyer, L. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, 364–369.

He, L.; Lewis, M.; and Zettlemoyer, L. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, 643–653.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR* abs/1508.01991.

Iyyer, M.; Boyd-Graber, J.; Claudino, L.; Socher, R.; and Daumé III, H. 2014. A neural network for factoid question answering over paragraphs. In *Empirical Methods in Natural Language Processing*.

Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *CoRR* abs/1705.03551.

Khot, T.; Sabharwal, A.; and Clark, P. 2017. Answering complex questions using open information extraction. *CoRR* abs/1704.05572.

Kwiatkowski, T.; Choi, E.; Artzi, Y.; and Zettlemoyer, L. S. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1545–1556.

Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, 282–289. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Liang, C.; Hsu, K.; Huang, C.; Li, C.; Miao, S.; and Su, K. 2016. A tag-based statistical english math word problem solver with understanding, reasoning and explanation. In *IJCAI*, 4254–4255. IJCAI/AAAI Press.

Liang, P. S. 2011. *Learning Dependency-Based Compositional Semantics*. Ph.D. Dissertation, University of California, Berkeley.

Lukovnikov, D.; Fischer, A.; Lehmann, J.; and Auer, S. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, 1211–1220. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Mrksic, N.; Séaghdha, D. Ó.; Thomson, B.; Gasic, M.; Rojas-Barahona, L. M.; Su, P.; Vandyke, D.; Wen, T.; and Young, S. J. 2016. Counter-fitting word vectors to linguistic constraints. 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*, 142–148.

Nogueira, R., and Cho, K. 2017. Task-oriented query reformulation with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, 574–583.

Palmer, M.; Gildea, D.; and Kingsbury, P. 2005. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.* 31(1):71–106.

Qiu, X., and Huang, X. 2015. Convolutional neural tensor network architecture for community-based question answering. In *IJCAI*, 1305–1311.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Reddy, S.; Täckström, O.; Collins, M.; Kwiatkowski, T.; Das, D.; Steedman, M.; and Lapata, M. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics* 4:127–140.

Sachan, M.; Dubey, K.; and Xing, E. P. 2016. Science question answering using instructional materials. In *ACL (2)*. The Association for Computer Linguistics.

Saha, D.; Floratou, A.; Sankaranarayanan, K.; Minhas, U. F.; Mittal, A. R.; and Özcan, F. 2016. Athena: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment* 9(12):1209–1220.

Seo, M. J.; Hajishirzi, H.; Farhadi, A.; Etzioni, O.; and Malcolm, C. 2015. Solving geometry problems: Combining text and diagram interpretation. In *EMNLP*, 1466–1476. The Association for Computational Linguistics.

Shen, Y.; Rong, W.; Jiang, N.; Peng, B.; Tang, J.; and Xiong, Z. 2015. Word embedding based correlation model for question/answer matching. *arXiv preprint arXiv:1511.04646*.

Sun, H.; Ma, H.; Yih, W.-t.; Tsai, C.-T.; Liu, J.; and Chang, M.-W. 2015. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, 1045–1055. New York, NY, USA: ACM.

Tan, M.; Xiang, B.; and Zhou, B. 2015. Lstm-based deep learning models for non-factoid answer selection. *CoRR* abs/1511.04108.

Trischler, A.; Wang, T.; Yuan, X.; Harris, J.; Sordoni, A.; Bachman, P.; and Suleman, K. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Trivedi, P.; Maheshwari, G.; Dubey, M.; and Lehmann, J. 2017. A corpus for complex question answering over knowledge graphs. In *Proceedings of 16th International Semantic Web Conference - Resources Track (ISWC'2017)*.

Vtyurina, A., and Clarke, C. L. 2016. Complex questions:let me google it for you. In *Proceedings of the second Web QA Workshop WEBQA 2016*.

Wang, D., and Nyberg, E. 2016. Mu oaqa at trec 2016 liveqa: An attentional neural encoder-decoder approach for answer rankin. In *Proceedings of The Twenty-Fifth Text REtrieval Conference, TREC 2016*.

Xu, K.; Reddy, S.; Feng, Y.; Huang, S.; and Zhao, D. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *Proceedings of the Association for Computational Linguistics (ACL 2016)*. Berlin, Germany: Association for Computational Linguistics.

Yang, B., and Mitchell, T. M. 2017. A joint sequential and relational model for frame-semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, 1247–1256.

Yih, W.; Chang, M.; He, X.; and Gao, J. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL (1)*, 1321–1331. The Association for Computer Linguistics.

Yin, P.; Duan, N.; Kao, B.; Bao, J.; and Zhou, M. 2015. Answering questions with complex semantic constraints on open knowledge bases. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, 1301–1310. New York, NY, USA: ACM.

Zettlemoyer, L. S. 2009. *Learning to map sentences to logical form*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Zhang, K.; Wu, W.; Wang, F.; Zhou, M.; and Li, Z. 2016. Learning distributed representations of data in community question answering for question retrieval. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, 533–542. New York, NY, USA: ACM.

Zhou, J., and Xu, W. 2015. End-to-end learning of semantic role labeling using recurrent 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)*, volume 1, 1127–1137.

Specific Question Generation for Reading Comprehension

Atsushi Otsuka, Kyosuke Nishida, Itsumi Saito, Hisako Asano, Junji Tomita
NTT Media Intelligence Laboratories, NTT Corporation
1-1 Hikarinooka, Yokosuka, Kanagawa, 239-0847, Japan
{atsushi.otsuka.vs, kyosuke.nishida.rx, itsumi.saito.df, hisako.asano.fe, junji.tomita.xa}@hco.ntt.co.jp

Abstract

We investigated specific question generation from a question about a passage in a reading comprehension task. In contrast to previous question reformulation work, our goal here is to specify ambiguous question intent with the given passage. In this new task, a system proposes several specific question candidates so that users can choose the question that is closest to their intent. For this, we propose an end-to-end neural network model that combines copy mechanisms with an attentional encoder-decoder. It locates missing information from the given passage to generate multiple specific questions. We also introduce a sentence compression method in order to create a training corpus for our model from existing reading comprehension datasets. Experimental results with the SQuAD dataset demonstrated that our model generated specific questions that can improve reading comprehension accuracy.

Introduction

The importance of question answering has increased with the development of various smart interactive devices such as smart speakers. In the research field of question answering, *reading comprehension*, a technology to read a passage and then answer a question about it, has particularly attracted the attention of many researchers due to its high accuracy and broad range of applications. Recently, a variety of large-scale datasets for reading comprehension have been released. For example, the Stanford Question Answering Dataset (SQuAD) is a set of Wikipedia articles and questions posed by crowdsourcing (Rajpurkar et al. 2016). Many neural network-based reading comprehension systems such as BiDAF (Seo et al. 2017) and QANet (Yu et al. 2018) have been proposed and brought about significant progress to the point that the reading comprehension performance is now comparable to that of humans.

However, there are still many problems when it comes to implementing this technology. In this study, we focused on reading comprehension when given ambiguous questions. There are many situations in which questioners will ask ambiguous questions in the real world. Take an interaction between questioners and operators of a contact center as an example, where the questioner does not always ask the operator a specific question. The operator supplements the question while predicting the intention of the questioner with knowledge documents such as answer books or manuals and queries the user to confirm it. Even in an automatic question answering system utilizing reading comprehension, users do not always input a specific question that the system can answer because the users do not know about the knowledge texts referred by the reading comprehension system. Thus, we feel that a function to supplement the question, similar to what is done by contact center operators, is necessary for reading comprehension systems.

In this paper, we tackle a novel task called *Specific Question Generation* (SQG; Fig. 1). SQG specifically revises the input question and suggests several *specific question* (SQ)

Passage: CBS provided digital streams of the game via CBSSports.com, and the CBS Sports apps on *tablets, Windows 10, Xbox One and other digital media players* (such as Chromecast and Roku). Due to Verizon Communications exclusivity, streaming on smartphones was only provided to *Verizon Wireless customers* via the NFL Mobile service. The ESPN Deportes Spanish broadcast was made available through WatchESPN.

Question: What app did viewers use to watch the game on their smartphones?
Answer of worker #1: NFL Mobile service
Answer of worker #2: the CBS Sports apps
Answer of worker #3: NFL

Specific Question #1: What app did viewers use to watch the game on their smartphones *of Verizon Wireless?*
Coressponding system answer: NFL Mobile service

Specific Question #2: What app did viewers use to watch the game on their *tablets?*
Coressponding system answer: CBS Sports apps

Figure 1: Specific question generation task. In this sample in SQuAD, answers were not unanimous even among human workers. Our goal is to generate questions that have specific correlation with the given passage so that users can choose the question that is closest to their intent.

candidates so that users are able to choose the SQ that is closest to their intent and obtain a highly accurate answer from the reading comprehension.

We also propose a *Specific Question Generation Model* (SQGM) for facilitating this task. Our model is based on an encoder-decoder model and uses two copy mechanisms (question copy and passage copy). The key idea is that the missing information in the user-input question is described in the passage. The decoder with the copy mechanisms generates specific questions by locating the missing information from the given passage (passage copy) while retaining the main part of the input question (question copy). In addition, we create a training dataset from reading comprehension datasets by generating pseudo ambiguous questions by means of sentence compression. Our main contributions are as follows:

- We designed an end-to-end specific question generation model based on a deep neural network that features an encoder-decoder and two copy mechanisms.
- We proposed a method of creating a training corpus for the model consisting of ambiguous and specific sentences by means of sentence compression.
- We demonstrated with SQuAD that our model generated specific questions that can improve the accuracy of the pre-trained BiDAF model (Gardner et al. 2018).

Problem Statement

In this section, we state the specific question generation task and provide definitions.

PROBLEM 1 (SPECIFIC QUESTION GENERATION: SQG). When given a question and a passage, the system suggests several *specific question* (SQ) candidates so that users can choose the SQ that is closest to their intent.

Definition 1 (SPECIFIC QUESTION GENERATION MODEL: SQGM). SQGM is a neural network model to realize SQG. When a question and a passage are given, the model generates and outputs specific questions.

Definition 2. A *question*, $q = \{q_1, ..., q_J\}$, is given a sentence in natural language where $q_1, ..., q_J$ are tokenized words represented by one-hot vectors.

Definition 3. A *passage*, $x = \{x_1, ..., x_T\}$, is a short part of a document in natural language. In contains an answer to the question and does not include any non-textual information such as images.

Definition 4. A *specific question (SQ)*, $y = \{y_1, ..., y_K\}$, is a natural language sentence generated by the specific generation question model. It is a specified question based on the content of the passage.

Method

The proposed specific question generation model, SQGM, is based on an attentional encoder-decoder model (Bahdanau, Cho, and Bengio 2015) and combines two copy mechanisms (Cao et al. 2017) with the encoder-decoder model.

Our model consists of three layers (Fig. 2):

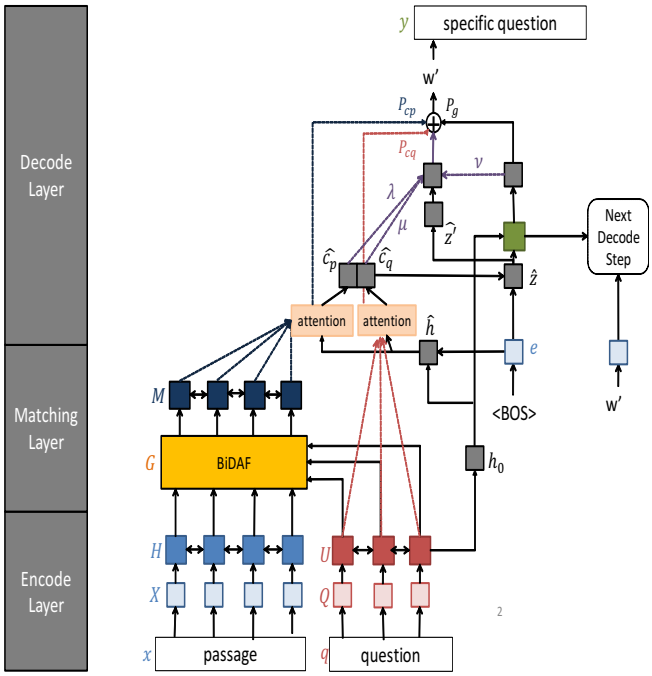


Figure 2: Proposed specific question generation model (SQGM).

1. The **encoding layer** maps all passage and question words into a vector space by a pre-trained word embedding model and encodes the temporal interactions between words.
2. The **matching layer** captures the interaction among passage and question words and produces a sequence of question-aware passage word vectors.
3. The **decoding layer** generates words in order to create specific questions with an attentional RNN decoder and two copy mechanisms.

Encoding layer

This layer first projects one-hot vectors (size: V) x and q into a v-dimensional continuous vector with a pre-trained word embedding matrix $W_e \in \mathbb{R}^{v \times V}$. The embedded vectors are passed to a highway network (Srivastava, Greff, and Schmidhuber 2015), and it outputs two sequences of vectors: $X \in \mathbb{R}^{v \times T}$ and $Q \in \mathbb{R}^{v \times J}$.

The passage and question vectors are then entered into recurrent neural networks (RNNs). We utilize a single-layer GRU (Cho et al. 2014) that is shared by the question and the passage. The GRU has both forward and backward directions (Bi-GRU) and concatenates the d-dimensional outputs of the two GRUs. Finally, it obtains contextual sequence matrix $H \in \mathbb{R}^{2d \times T}$ from X and $U \in \mathbb{R}^{2d \times J}$ from Q.

The decoding layer requires an initial state from the encoder. The initial state vector $h_0 \in \mathbb{R}^{2d}$ is created by a self-attention in the question contextual sequence matrix U, as

$$h_0 = \sum_{j<J} \alpha_j U_j, \quad (1)$$

where $\alpha_j = \mathrm{softmax}_j(U_J^\top U_j)$ and $U_J^\top$ denotes a transposed final state vector of the question contextual sequence.

Matching layer

This layer identifies a correspondence relationship between the passage and the question. A bi-directional attention flow (BiDAF) (Seo et al. 2017) mechanism is utilized for this layer. The BiDAF computes attentions in two directions to fuse information from the passage to the question words as well as from the question to the passage and creates new passage contextual vectors with a dependency on the question.

The BiDAF first computes a similarity matrix $S \in \mathbb{R}^{T \times J}$. The similarity between the t-th passage word and the j-th question word S_{tj} is

$$S_{tj} = w_s^\top [H_t; U_j; H_t \odot U_j], \tag{2}$$

where $w_s \in \mathbb{R}^{6d}$ are learnable parameters, $[;]$ denotes a vector concatenation across a row, and $\odot$ is an element-wise product. Next, we obtain a passage-to-question attention and question-to-passage attention from the similarity matrix.

Passage-to-question attention signifies which question words are most relevant to each passage word. The attention vector that denotes the t-th passage word $\check{U}_t \in \mathbb{R}^{2d}$ is computed as follows:

$$\check{U}_t = \textstyle\sum_j a_{tj} U_j, \tag{3}$$

where $a_t = \mathrm{softmax}_j(S_t)$.

Question-to-passage attention signifies which passage words have the closest similarity to one of the question words. It obtains the passage contextual vector $\check{h} \in \mathbb{R}^{2d}$ as follows:

$$\check{h} = \textstyle\sum_t b_t H_t, \tag{4}$$

where $b = \mathrm{softmax}_t(\max_j(S))$.

Finally, it obtains $\check{H} \in \mathbb{R}^{2d \times T}$ by tiling the vector T times across the columns.

Bi-directional attention computes G to obtain question-aware vectors of each passage word, as

$$G = [H; \check{U}; H \odot \check{U}; H \odot \check{H}] \in \mathbb{R}^{8d \times T}. \tag{5}$$

The matching layer also uses a single layer bi-GRU and obtains $M \in \mathbb{R}^{2d \times T}$ from G to capture the interaction among passage words conditioned on the question.

Decoding layer

This layer outputs a specific question. Our decoder consists of an RNN-based language model and passage and question copy mechanisms.

Generative language modeling This layer uses a single-layer GRU with an attention mechanism and a softmax layer for language modeling.

Let $y = \{y_1, ...\}$ be the output word sequence of a specific question. This generative model outputs a distribution over a vocabulary of fixed-size V_g:

$$P_g(y_{s+1}|y_{\leq s}, X, Q) = \mathrm{softmax}(W_g h_{s+1} + b_g), \tag{6}$$

where $W_g \in \mathbb{R}^{2d \times V_g}$ and $b_g \in \mathbb{R}^{V_g}$ denote learnable parameters and $h_s \in \mathbb{R}^{2d}$ is the s-th hidden state of the GRU. h_s is updated as follows:

$$h_{s+1} \leftarrow \mathrm{GRU}(h_s, \hat{z}_s), \tag{7}$$

where $\hat{z}_s$ is the input vector to the GRU.

This model obtains $\hat{z}_s$ as follows. First, it takes the output word of the decoder in the preceding step y_s as input (the first word is the beginning token $< BOS >$) and projects y_s into embedding vector $e_s \in \mathbb{R}^v$, as in the encoding layer. Next, it utilizes an attention mechanism by using the hidden states as a query. An attention query vector $\hat{h}_s \in \mathbb{R}^{2d}$ is created from the embedding vector e_s and the hidden state h_s:

$$\hat{h}_s = f(W_d[e_s; h_s] + b_d). \tag{8}$$

The model calculates passage attention $\alpha_{st} \in \mathbb{R}^T$ and question attention $\beta_{sj} \in \mathbb{R}^J$ vectors based on the attention query vector:

$$\alpha_{st} = \mathrm{softmax}_t(M_t \cdot \hat{h}_s), \tag{9}$$

$$\beta_{sj} = \mathrm{softmax}_j(U_j \cdot \hat{h}_s). \tag{10}$$

Finally, we calculate the s-th input vector of the GRU $z_s \in \mathbb{R}^{v+4d}$ as

$$\hat{c}_{ps} = \textstyle\sum_t \alpha_{st} M_t, \tag{11}$$

$$\hat{c}_{qs} = \textstyle\sum_j \beta_{sj} U_j, \tag{12}$$

$$\hat{z}_s = [e_s; \hat{c}_{ps}; \hat{c}_{qs}], \tag{13}$$

where $W_d \in \mathbb{R}^{(v+2d) \times 2d}$ and $b_d \in \mathbb{R}^{2d}$ are learnable parameters and f denotes a ReLU function as an activation faction.

Copy mechanism Our model uses a copy mechanism based on (Cao et al. 2017) to extract the most useful word for specific questions from a passage or a question. This mechanism utilizes the attention weights of the language model as the occurrence probabilities of passage or question words. Hence, the outputs of the coping decoder are calculated as follows:

$$P_{cp}(y_{s+1}|y_{\leq s}, X, Q) = \sum_t \mathbb{1}(y_{s+1} = X_t)\alpha_{(s+1)t}, \tag{14}$$

$$P_{cq}(y_{s+1}|y_{\leq s}, X, Q) = \sum_j \mathbb{1}(y_{s+1} = Q_j)\beta_{(s+1)j}, \tag{15}$$

where $\mathbb{1}(y_s = X_t)$ denotes 1 if $y_s = X_t$ and otherwise 0. In the same way, $\mathbb{1}(y_s = Q_j)$ denotes 1 if $y_s = Q_j$ and otherwise 0.

Weighted linear combination of generative language and copy models Finally, the decoding layer outputs word occurrence probabilities $P(y_{s+1}|y_{\leq s}, X, Q)$ and decides the output word y_{s+1} by referring to the maximum value element of the probabilities,

$$\begin{aligned} P(y_{s+1}|y_{\leq s}, X, Q) = {} & \lambda_s P_g(y_{s+1}|y_{\leq s}, X, Q) \\ & + \mu_s P_{cp}(y_{s+1}|y_{\leq s}, X, Q) \\ & + \nu_s P_{cq}(y_{s+1}|y_{\leq s}, X, Q), \end{aligned} \tag{16}$$

where λ, μ, and ν are learnable weighting factors having the constraints $\lambda, \mu, \nu \in [0, 1]$ and $\lambda + \mu + \nu = 1$. These parameters are computed as

$$\hat{z}'_s = \mathrm{Highway}_2(\hat{z}_s), \tag{17}$$

$$\gamma_{sk} = \mathrm{softmax}(W_c \hat{z}'_s + b_c), \tag{18}$$

$$\lambda_s = \gamma_{s0}, \quad \mu_s = \gamma_{s1}, \quad \nu_s = \gamma_{s2},$$

where $\mathrm{Highway}_2()$ denotes a two-layer highway network and $W_c \in \mathbb{R}^{(v+4d)\times 3}$ and $b_c \in \mathbb{R}^3$ are learnable parameters.

Model training

We compute the loss L by a negative log likelihood as

$$L = -\frac{1}{N}\sum_i \sum_s \log P(y^{(i)}_{s+1}|y^{(i)}_{\leq s}, X^{(i)}, Q^{(i)}), \qquad (19)$$

where N is a size of a mini-batch and i is the index of the sample in the mini-batch.

Creation of training corpus

Our model requires a training corpus that contains passages, questions, and specific questions in order to optimize the model parameters. However, creating such a training corpus from scratch is costly.

We propose a method of creating a training corpus by using existing reading comprehension datasets (such as SQuAD) and applying sentence compression. We regard the questions included in the dataset as specific questions. Then, we create shorter questions from the specific questions by means of sentence compression and train our model to generate the specific questions when it takes the passage and the short question as inputs.

The sentence compression generates a shorter paraphrase of a sentence. This is a standard NLP task and various ways of using it have been proposed (Wang et al. 2017; Hasegawa et al. 2017). In this work, we utilize an unsupervised sentence compression method based on dependency parsing and integer programming (IP). This method first creates a parse tree by dependency parsing and identifies trimmable nodes by IP. When given a sentence that has L words, our sentence compression is formulated as follows:

$$\begin{aligned} \underset{a_i\in\{0,1\},1\leq i\leq L}{\textbf{maximize}} \quad & \textstyle\sum_{i\leq L} w_i a_i \\ \textbf{subject to} \quad & \textstyle\sum_{i\leq L} a_i \leq l \\ & a_{\mathrm{parent}(i)} - a_i \geq 0 \ \ (1 \leq i \leq L) \\ & a_{\mathrm{argmax}(w_i)} = 0 \end{aligned} \qquad (20)$$

where a_i is 1 if the i-th word is selected in the compressed sentence and 0 otherwise, w_i denotes the importance of the i-th word, and l is a maximum length of a compressed sentence. $a_{parent(i)}$ denotes the word of a parent node corresponding to a child word w_i on the dependency tree.

First, the constraint $\sum_{i\leq L} a_i \leq l$ controls the length of the compressed sentence. We create compressed questions while changing l to increase the amount of generated data. Next, the constraint $a_{parent(i)} - a_i \geq 0$ forbids the selection of a child node without also selecting its parent node. The constraint $a_{\mathrm{argmax}(w_i)} = 0$ is newly added into the general sentence compression formulation. The general formulation retains the words of the input sentence in accordance with the word importance scores, while the goal of our sentence compression is to create questions that lack important information to answer. Thus, we remove the most important word in the sentence compression.

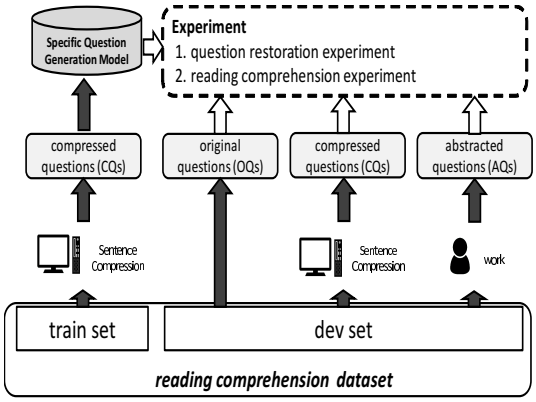


Figure 3: Overview of dataset for experiment.

The word importance is calculated on the basis of word occurrences in the training corpus as $w_i = \log F/F(w_i)$, where $F(w_i)$ denotes the word w_i frequency and F is the total number of times a word appears in the corpus.

Experiments

In this section, we describe the experiments used to evaluate our proposed method. We conducted two experiments for the evaluation: a question restoration experiment and an evaluation of the reading comprehension. First, we explain the dataset and model configurations for the experiments. Next, we describe the experiment configurations of both experiments. Finally, we present the experiment results.

Dataset

We used the SQuAD 1.1 (Rajpurkar et al. 2016) and NewsQA (Trischler et al. 2017) datasets. SQuAD is one of the most basic reading comprehension datasets created from Wikipedia articles. NewsQA is created from news articles and consists of shorter and less specific questions than SQuAD. We prepared the following three types of input questions for our specific question generation model.

1. **Original questions (OQs)** are original questions with no processing.

2. **Compressed questions (CQs)** are short questions created automatically from the original questions by the sentence compression.

3. **Abstracted questions (AQs)** are short questions created manually from the original questions. A worker made the AQs as short as possible while retaining the question intention of the original. AQs were only created for the SQuAD dataset.

Figure 3 shows an overview of the datasets. The statistics of the datasets are listed in Table 1. We created the CQs by sentence compression while changing the maximum sentence length, l, from 3 to $L-2$ (L is the length of the input question). To create CQs, we used all of the original questions in the training set and used a maximum of three original questions (randomly selected) for each passage in the development set. The AQs were created from 350 original

Table 1: Reading comprehension datasets used in the experiments. Note that N denotes the number of passages and questions and L denotes the mean length (in tokens).

	train		dev		
	OQs	CQs	OQs	CQs	AQs
SQuAD					
N. passages	18,896	18,896	2,067	2,067	278
N. questions	87,599	120,000	10,570	32,208	350
L. passage	142.8	142.8	144.5	144.5	140.8
L. question	13.7	7.9	13.3	7.7	6.6
NewsQA					
N. passages	3,072	3,072	1,900	1,900	–
N. questions	10,510	69,959	3,000	16,569	–
L. passage	237.9	237.9	297.5	144.5	–
L. question	10.4	6.3	10.7	6.3	–

questions randomly selected from the SQuAD development set. Examples of compressed and abstracted questions can be found in the *supplemental material*.

Model configuration

Preprocessing We used the spaCy tokenizer[1] for all model training and experiments and all of the words are converted to lowercase. Our model used pre-trained 300-d fastText (Bojanowski et al. 2017) embeddings trained with full Wikipedia articles and all of the passages and questions in the training set of SQuAD.

Training process We used the same configuration for all datasets. Models were trained with two GPUs (Quadro P6000). Each GPU processed a minibatch of size 96. The hidden size d was set to 128. The generative vocabulary size in the decoder layer V_g was set to 5,000 (it covers over 90% of all of vocabularies for the questions in SQuAD). A dropout (Srivastava et al. 2014) rate of 0.2 was used for input layers, and 0.5 was used for the internal layers in highway networks. We used the Adam optimizer. The number of epochs was 5 (for a total of 64,300 training steps). A beam size of 5 was used for the beam search in the decoder.

Question restoration experiment

The question restoration experiment evaluates the quality of specific question generation. In the experiments with SQuAD and NewsQA, each model generated the best specific question (SQ) for each compressed question (CQ) or abstracted question (AQ). We used original questions (OQs) as ground-truths.

Comparison model We prepared five models for the question restore experiments, including our proposed model (**proposed**). We first created a model from the proposed model that contains no copy mechanisms (**w/o copy**). Next, we removed the copy mechanism from the passage (**w/o p_copy)** or the question (**w/o q_copy**). Finally, we created a model by removing the BiDAF mechanism (**w/o BiDAF**).

[1]https://spacy.io/

Table 2: Results of question restoration experiments in the SQuAD dataset.

	Model	Precision	Recall	F1	Sim
SQuAD	CQs	1.00	0.586	0.715	0.758
	: w/o copy	0.214	0.203	0.205	0.256
	: w/o p_copy	0.346	0.301	0.319	0.361
	: w/o q_copy	0.669	0.539	0.619	0.770
	: w/o BiDAF	**0.856**	0.678	0.747	0.814
	: **Proposed**	0.824	**0.718**	**0.758**	**0.834**
	AQs	1.00	0.626	0.754	0.794
	: w/o copy	0.265	0.282	0.260	0.345
	: w/o p_copy	0.365	0.363	0.353	0.436
	: w/o q_copy	0.707	0.570	0.647	0.621
	: w/o BiDAF	**0.900**	0.650	0.735	0.810
	: **Proposed**	0.844	**0.716**	**0.759**	0.**837**
NewsQA	CQs	1.00	0.581	0.715	0.715
	: w/o copy	0.281	0.231	0.250	0.232
	: w/o p_copy	0.702	0.597	0.637	0.695
	: w/o q_copy	0.623	0.541	0.568	0.658
	: w/o BiDAF	**0.832**	0.631	0.711	0.738
	: **Proposed**	0.805	**0.690**	**0.731**	**0.766**

This model used an output of the encoding layer, H, instead of the output of the matching layer, M.

Evaluation metrics We used the following metrics in the question restoration experiment. **Precision** and **Recall** are calculated between a generated SQ and the ground-truth (original) question at the word-level. **F1 score** is the harmonic average of the precision and recall. **Semantic similarity** is a relevance score between SQ and OQ. It is calculated by a cosine distance between two semantic vectors. The vectors are encoded from questions by the universal sentence encoder (Cer et al. 2018).

Results Table 2 shows the results on the SQuAD and NewsQA datasets. The results of the CQs and AQs in the 'Model' column denote the scores between the input and original questions.

Our proposed model statistically significantly outperformed all comparative models for all the scores other than precision (t-test; $p < .01$). Although our specific generation model was trained with compressed questions automatically created by sentence compression, it also performed well when questions abstracted by hand were given.

The results of ablation tests demonstrated the effectiveness of both the question and passage copy mechanisms. In particular, the passage copy mechanism strongly contributed to the performance in the SQuAD dataset. SQuAD contains questions that are longer and more specific than those in NewsQA; thus, the passage copy mechanism could learn from SQuAD correctly. Also, we confirmed that the question copy mechanism contributed to the performance: specifically, it was effective for retaining the main part of input question intent. In fact, approximately 61% of the words of specific questions generated were copied from the questions (21% of words were copied from the passages and 19%

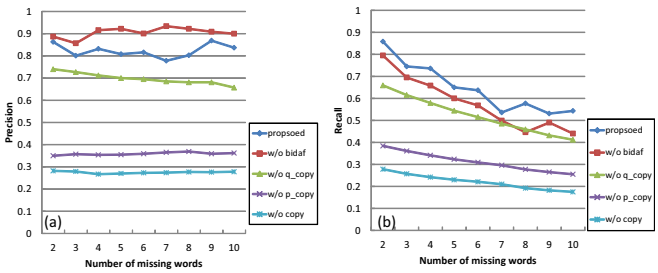


Figure 4: (a) Precision and (b) recall due to number of missing words in the question restoration experiment on the SQuAD dataset.

were generated). The model without the BiDAF mechanism achieved the highest score in precision. This model did not capture the interaction between a question and a passage, and thus was less aggressive in specifying question intent with the passage information than our model. Recall, F1, and semantic similarity scores were more important to evaluate the capability of finding missing information in the input question from the passage.

Figure 4 shows the precision and recall changes when changing the number of missing words in sentence compression. We can see that our model restored original questions accurately with the given passage even for strongly compressed questions.

Reading comprehension experiment with specific questions

In this experiment with the SQuAD dataset, we evaluated the effectiveness of the specific questions generated by our model for improving reading comprehension. We compared the reading comprehension accuracies corresponding to the specific questions (SQs) with those corresponding to the original, compressed, and abstracted questions.

Reading comprehension model We used the AllenNLP BiDAF model trained with SQuAD (Gardner et al. 2018), which is available on the Web[2].

Evaluation metrics We compared the answers from the BiDAF for each question with the ground-truth answers on the basis of three types of exact-match metrics, where the exact-match measures the percentage of predictions that match the ground-truth exactly. We used a beam search to generate five specific questions (SQ1 to SQ5, in descending order of the occurrence probability in decoding). The first metric, for evaluating the performance in the setting of Problem 1, is **Success@k**, which is the percentage of times that the answers from BiDAF for the input question and the top-k SQs contained the ground-truths. The second one, for evaluating the individual contribution of SQ, is the individual **accuracy**, which is the accuracy of the answers from BiDAF for each rank of SQs. The third one is the individual **improvement**, which is the individual accuracy when the answers from BiDAF for the input question were wrong.

[2]http://allennlp.org/models

Results Table 3 shows the results when given the original questions as input. Reading comprehension was improved up to 7.6% by in Success@5. This result demonstrates that users can choose the question that is closest to their intent from SQs and obtain an accurate answer from the chosen SQ. Interestingly, SQ1 was the best in the individual accuracy, while SQ4 was the best in the individual improvement. This indicates that top SQs were semantically close to the original intent and other SQs covered different intents.

When given the compressed questions as input, reading comprehension was improved up to 22.4% in Success@5. In this case, where input questions were highly ambiguous, SQ1 was the best in both the individual accuracy and improvement metrics.

When given the abstracted questions as input, reading comprehension was improved up to 14.4% in Success@5. This result exhibits a similar tendency with the result when the original questions were used as input. On the other hand, the improvement scores, when the reading comprehension model returned wrong answers for the input questions, were higher compared to the original questions. We can see from these results that our model improved the accuracy of reading comprehension when given ambiguous questions.

Figure 5 shows an example of specific question generation. In this example, the passage describes the former and new head coaches of the Denver Broncos, and the BiDAF incorrectly answered the name of the former head coach for the question "Who is the head coach of the broncos?". We can see that SQ4 and SQ5 specified the original question intent from "head coach" to "new head coach" with the given passage, and BiDAF can obtain the correct answer with the SQs. We should note that our model generated paraphrases of the input questions such as in SQ3, where the answer from BiDAF for SQ3 was correct. The improvements in Table 3 include the results of such paraphrasing or grammatical error correction in addition to the question specifying.

Related work and discussion

Question reformulation Question reformulation has been used to improve question answering accuracy. Recently, Buck et al. proposed a reinforcement learning method for reformulating questions to elicit the best possible answers (Buck et al. 2018). Their method uses a black-box and a fixed QA system that gives the target performance metric and trains end-to-end neural networks to maximize answer quality. Their method takes only a question as input to reform the question while retaining the original intent, while our method takes a question and a passage as inputs to specify question intent with the passage.

Moreover, query reformulation techniques have been well studied in conventional IR-based QA systems. The systems have reformulated queries to enable their IR method to cover many textual variants. Such reformulation is dependent on the redundancy of the knowledge source (Brill, Dumais, and Banko 2002; Lin 2007) and does not work well on reading comprehension.

Table 3: Results of reading comprehension experiments with specific questions (SQs) generated from original questions (OQs) in SQuAD.

	OQs			CQs			AQs		
	Success@k	Accuracy	Improve	Success@k	Accuracy	Improve	Success@k	Accuracy	Improve
Input	0.645	0.645	–	0.347	0.347	–	0.445	0.445	–
SQ1	0.663	**0.595**	0.0520	0.482	**0.389**	**0.207**	0.501	**0.455**	0.108
SQ2	0.684	0.555	0.0811	0.519	0.374	0.194	0.542	0.422	0.134
SQ3	0.698	0.544	0.0869	0.541	0.363	0.189	0.571	0.402	0.134
SQ4	0.710	0.542	**0.100**	0.558	0.363	0.191	0.580	0.403	**0.160**
SQ5	**0.721**	0.522	0.0957	**0.571**	0.343	0.181	**0.589**	0.420	0.134

Following their loss in the divisional round of the previous season 's playoffs , the Denver Broncos underwent numerous coaching changes , including a mutual parting with head coach John Fox (who had won four divisional championships in his four years as Broncos head coach) , and the hiring of Gary Kubiak as the new head coach. (..)

Ground-truth answers: Gary Kubiak, Gary Kubiak, Kubiak

OQ: who is the head coach of the broncos ?

Original and specific questions (OQ, SQ)	Answers from BiDAF
SQ1 : who is the head coach of the denver broncos ?	John Fox
SQ2 : who is the head coach of the broncos ?	John Fox
SQ3: who is the head coach of the denver ?	John Fox
SQ4 : who is the new head coach of the broncos ?	Gary Kubiak
SQ5 : who is the new head coach of the denver broncos ?	Gary Kubiak

Copied from passage Copied from question Generated

Figure 5: An example of specific question generation. A part of the passage '(...)' is omitted due to space. The passage, ground-truth answers, and original question are from the SQuAD development set. The BiDAF used here is the implementation of AllenNLP (Gardner et al. 2018).

Question generation Question generation is the reverse task of reading comprehension and has potential for educational purposes and for providing question-answer pairs. Du, Shao, and Cardie proposed a method for generating a natural question related to information in a sentence including the answer (Du, Shao, and Cardie 2017). Duan et al. also proposed a method of question generation and used it to improve the answer sentence selection task (Duan et al. 2017). Moreover, question generation from other media has been studied, such as visual question generation (Mostafazadeh et al. 2016; Jain, Lazebnik, and Schwing 2018) and question generation from knowledge bases (Serban et al. 2016; ElSahar, Gravier, and Laforest 2018). The specific question generation task is different from this task in that a system is given a question as input.

Question paraphrasing Question paraphrasing is a kind of question reformulation. In a recent work by Dong et al., a question and its paraphrases are fed into a paraphrase scoring model and a QA model, and then the system decides the answer by using a weighted voting with the scores (Dong et al. 2017). General purpose methods and paraphrasing datasets have been studied extensively and can be applied to question paraphrasing (Gupta et al. 2018). The goal of question paraphrasing is to generate sentences that convey the same meaning using different wording, and so it cannot be used to specify the intent of an ambiguous question.

Relevance feedback and query expansion Relevance feedback and query expansion are conventional techniques used in information retrieval (Manning, Raghavan, and Schtze 2008). In relevance feedback, the system revises a user query with the documents that the user explicitly or implicitly marks as relevant. In contrast, in query expansion, the system suggests additional query terms without user feedback. Our specific question generation is more powerful in terms of understanding natural language without user feedback and a large number of query logs.

Conclusion

In this study, we tackled a new task: specific question generation from a question about a passage in reading comprehension. In contrast to previous work, our work makes it possible to specify the intent of an ambiguous question with the given passage. We believe this idea is novel and has a broad range of applications such as interactive question answering in which users can choose the question that is closest to their original intent. In this paper, we proposed an end-to-end specific question generation model and a training method for the model from existing reading comprehension datasets. Experimental results with SQuAD demonstrated promising results in generating specific questions that enable a pre-trained BiDAF model to find the correct answer.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL)* 5:135–146.

Brill, E.; Dumais, S. T.; and Banko, M. 2002. An analysis of the askmsr question-answering system. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Buck, C.; Bulian, J.; Ciaramita, M.; Gajewski, W.; Gesmundo, A.; Houlsby, N.; and Wang, W. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Cao, Z.; Luo, C.; Li, W.; and Li, S. 2017. Joint copying and restricted generation for paraphrase. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 3152–3158.

Cer, D.; Yang, Y.; Kong, S.; Hua, N.; Limtiaco, N.; John, R. S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; Sung, Y.; Strope, B.; and Kurzweil, R. 2018. Universal sentence encoder. *CoRR* abs/1803.11175.

Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734.

Dong, L.; Mallinson, J.; Reddy, S.; and Lapata, M. 2017. Learning to paraphrase for question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, 875–886.

Du, X.; Shao, J.; and Cardie, C. 2017. Learning to ask: Neural question generation for reading comprehension. In *Association for Computational Linguistics (ACL)*, 1342–1352.

Duan, N.; Tang, D.; Chen, P.; and Zhou, M. 2017. Question generation for question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, 866–874.

ElSahar, H.; Gravier, C.; and Laforest, F. 2018. Zero-shot question generation from knowledge graphs for unseen predicates and entity types. In *North American Association for Computational Linguistics (NAACL)*. 218-228.

Gardner, M.; Grus, J.; Neumann, M.; Tafjord, O.; Dasigi, P.; Liu, N. F.; Peters, M. E.; Schmitz, M.; and Zettlemoyer, L. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR* abs/1803.07640.

Gupta, A.; Agarwal, A.; Singh, P.; and Rai, P. 2018. A deep generative framework for paraphrase generation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Hasegawa, S.; Kikuchi, Y.; Takamura, H.; and Okumura, M. 2017. Japanese sentence compression with a large training dataset. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 281–286.

Jain, U.; Lazebnik, S.; and Schwing, A. G. 2018. Two can play this game: Visual dialog with discriminative question generation and answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lin, J. 2007. An exploration of the principles underlying redundancy based factoid question answering. *ACM Transactions on Information Systems (TOIS)* 25(2):6.

Manning, C. D.; Raghavan, P.; and Schtze, H. 2008. *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press.

Mostafazadeh, N.; Misra, I.; Devlin, J.; Mitchell, M.; He, X.; and Vanderwende, L. 2016. Generating natural questions about an image. In *Association for Computational Linguistics (ACL)*.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2383–2392.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*.

Serban, I. V.; García-Durán, A.; Gülçehre, Ç.; Ahn, S.; Chandar, S.; Courville, A. C.; and Bengio, Y. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Association for Computational Linguistics (ACL)*.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Highway networks. *CoRR* abs/1505.00387.

Trischler, A.; Wang, T.; Yuan, X.; Harris, J.; Sordoni, A.; Bachman, P.; and Suleman, K. 2017. NewsQA: A machine comprehension dataset. In *Workshop on Representation Learning for NLP (RepL4NLP@ACL)*, 191–200.

Wang, L.; Jiang, J.; Chieu, H. L.; Ong, C. H.; Song, D.; and Liao, L. 2017. Can syntax help? improving an lstm-based sentence compression model for new domains. In *Association for Computational Linguistics (ACL)*, 1385–1393.

Yu, A. W.; Dohan, D.; Luong, M.-T.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations (ICLR)*.

Translating Natural Language to SQL using Pointer-Generator Networks and How Decoding Order Matters

Denis Lukovnikov[1], Nilesh Chakraborty[1], Jens Lehmann[1,2], and Asja Fischer[3]
[1]Smart Data Analytics Group, University of Bonn, Bonn, Germany
[2]Enterprise Information Systems, Fraunhofer IAIS, St. Augustin, Germany
[3]Ruhr University Bochum, Bochum, Germany

Abstract

Translating natural language to SQL queries for table-based question answering is a challenging problem and has received significant attention from the research community. In this work, we extend a pointer-generator network and investigate how query decoding order matters in semantic parsing for SQL. Even though our model is a straightforward extension of a general-purpose pointer-generator, it outperforms early work for WikiSQL and remains competitive to concurrently introduced, more complex models. Moreover, we provide a deeper investigation of the potential "order-matters" problem due to having multiple correct decoding paths, and investigate the use of REINFORCE as well as a non-deterministic oracle in this context. [1]

Introduction

Semantic parsing, the task of converting Natural Language (NL) utterances to their representation in a formal language, is a fundamental problem in Natural Language Processing (NLP) and has important applications in Question Answering (QA) over structured data and robot instruction.

In this work, we focus on QA over tabular data, which attracted significant research efforts (Zhong et al. 2017; Xu et al. 2017; Yu et al. 2018; Huang et al. 2018; Haug et al. 2018; Wang et al. 2018a; 2018b; Shi et al. 2018; Krishnamurthy et al. 2017; Iyyer et al. 2017; Pasupat and Liang 2015). In this task, given a NL question and a table, the system must generate a query that will retrieve the correct answers for the question from the given table.

The model we use in this paper is a straightforward extension of pointer-generators, and yet outperforms early work and compares well against concurrently developed models. Concretely, we add simple LSTM-based column encoders, skip connections and constrained decoding, as elaborated later in the paper.

In order to use sequence decoders for generating queries, the queries must first be linearized to sequences which can be used to train a sequence decoder. However, when translating NL questions to SQL queries, as in many semantic parsing tasks, target queries can contain unordered elements, which results in multiple valid decoding paths. The particular ordering of the unordered elements in the decoding path used for supervision can affect the performance of the trained SEQ2SEQ models. We provide a deeper investigation of the potential "order-matters" problem in translating NL to SQL that has been raised by previous work. In this context, we also investigate training with a non-deterministic oracle (Goldberg and Nivre 2012) as well as training with REINFORCE, both of which explore different possible linearizations of the target queries, and show that the use of a non-deterministic oracle can be beneficial when the original supervision sequences are ordered inconsistently.

In the following, we first introduce the problem, then describe our model and the training procedure, present an experimental analysis, and conclude with a comparison to related work.

Queries, Trees and Linearizations

As an illustration of table-based QA, consider the natural language question

> "*How much L1 Cache can we get with an FSB speed of 800MHz and a clock speed of 1.0GHz?*"

This question should be mapped to the following SQL query

```
SELECT  L1_Cache
WHERE   FSB_Speed = 800 (Mhz)
   AND  Clock_Speed = 1.0 (GHz)
```

which will be executed over a table listing processors, the sizes of their caches, their clocks speeds etc. In the query representation format we use, the example SQL query will be represented as the following sequence of output tokens:

```
SELECT L1_Cache AGG0 WHERE COND
FSB_Speed OP0 VAL 800 ENDVAL COND
Clock_Speed OP0 VAL 1.0 ENDVAL ,
```

where `AGG0` is a dummy "no aggregator" token that is used to indicate that no real aggregator should be applied and `OP0` is the = (equality) operator. Other aggregators, like `SUM` and `COUNT`, and other operators, like $<$ (less than) are also available.

As illustrated in Figure 1, the SQL query can also be represented as a tree where the root node has two children: `SELECT` and `WHERE`. Note that the order of the two conditions appearing in the `WHERE` clause is arbitrary and does

[1]This is an updated version of our previous anonymous version (`https://openreview.net/forum?id=HJMoLws2z`) from May 2018.

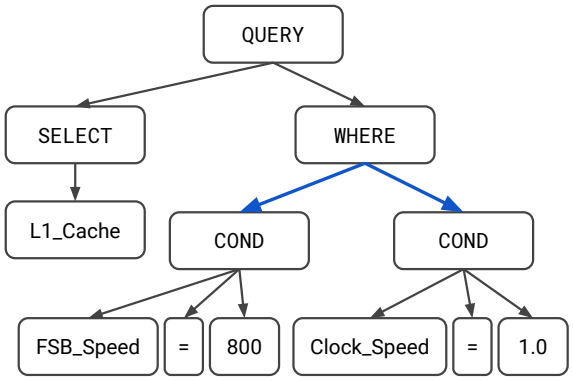


Figure 1: Example of a query tree. The blue arrows indicate unordered children.

not have any impact on the meaning of the query or the execution results. Trees containing such unordered nodes can be linearized into a sequence in different, equally valid, ways ("FSB Speed" first or "Clock Speed" first in the example, as illsustrated in Figure 2.). We refer to the linearization where the original order as given in the data set is preserved as the *original linearization*.

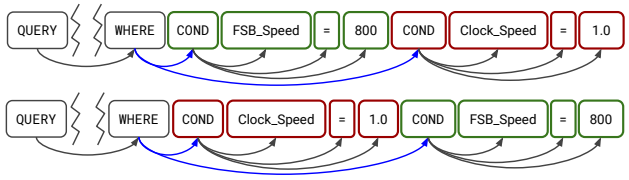


Figure 2: Two valid linearizations of the example query tree in Figure 1.

Model

We start from a sequence-to-sequence model with attention and extend the embedding and output layers to better suit the task of QA over tabular data. In particular, we use on-the-fly (Bahdanau et al. 2017) embeddings and output vectors for column tokens and implement a pointer-based (Gu et al. 2016; See et al. 2017) mechanism for copying tokens from the question. The resulting model is a Pointer-Generator (Gu et al. 2016; See et al. 2017) with on-the-fly representations for a subset of its vocabulary.

The Seq2Seq Model

The general architecture of our model follows the attention-based sequence-to-sequence (SEQ2SEQ) architecture. The following formally introduces the major parts of our SEQ2SEQ model. Details about the embedding and the output layers are further elaborated in later sections.

The SEQ2SEQ model consists of an encoder, a decoder, and an attention mechanism.

Encoder We are given a question $Q = [q_0, q_1, \ldots, q_N]$ consisting of NL tokens q_i from the set $\mathcal{V}^E$ (i.e., the encoder vocabulary). The tokens are first passed through an embedding layer that maps every token q_i to its vector representation $\mathbf{q}_i = W^E \cdot \mathsf{one\_hot}(q_i)$ where $W^E \in \mathbb{R}^{|\mathcal{V}^E| \times d^{emb}}$ is a learnable weight matrix and $\mathsf{one\_hot}(\cdot)$ maps a token to its one-hot vector.

Given the token embeddings, a bidirectional multi-layered Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) encoder produces the hidden state vectors $[\mathbf{h^*}_0, \mathbf{h^*}_1, \ldots, \mathbf{h^*}_N] = \mathsf{BiLSTM}([\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_N])$.

The encoder also contains skip connections that add word embeddings $\mathbf{q}_i$ to the hidden states $\mathbf{h^*}_i$.

Decoder The decoder produces a sequence of output tokens s_t from an output vocabulary $\mathcal{V}^D$ conditioned on the input sequence Q. It is realized by a uni-directional multi-layered LSTM. First, the previous output token s_{t-1} is mapped to its vector representation using the embedding function $\mathrm{EMB}(\cdot)$. The embeddings are fed to a BiLSTM-based decoder and its output states are used to compute the output probabilities over $\mathcal{V}^D$ using the output function $\mathrm{OUT}(\cdot)$. $\mathrm{EMB}(\cdot)$ and $\mathrm{OUT}(\cdot)$ are described in the following sections.

Attention We use attention (Bahdanau et al. 2014) to compute the context vector $\hat{\mathbf{h}}_t$, that is

$$a_i^{(t)} = \mathbf{h}_i \cdot \mathbf{y}_t \ , \tag{1}$$

$$\alpha_i^{(t)} = \mathsf{softmax}(a_0^{(t)}, \ldots, a_i^{(t)}, \ldots, a_N^{(t)})_i \ , \tag{2}$$

$$\hat{\mathbf{h}}_t = \sum_{i=0}^{N} \alpha_i^{(t)} \mathbf{h}_i \ , \tag{3}$$

where $\mathsf{softmax}(\cdot)_\mathsf{i}$ denotes the i-ith element of the output of the softmax function, $\mathbf{y}_t$ is the output state of the decoder and $\mathbf{h}_1, \ldots, \mathbf{h}_N$ are the embedding vectors returned by the encoder.

Embedding Function of the Decoder

The whole output vocabulary $\mathcal{V}^D$ can be grouped in three parts: (1) SQL tokens from $\mathcal{V}^{\mathrm{SQL}}$, (2) column ids from $\mathcal{V}^{\mathrm{COL}}$, and (3) input words from the encoder vocabulary $\mathcal{V}^E$, that is, $\mathcal{V}^D = \mathcal{V}^{\mathrm{SQL}} \cup \mathcal{V}^{\mathrm{COL}} \cup \mathcal{V}^E$. In the following paragraphs, we describe how each of the three types of tokens is embedded in the decoder.

SQL tokens: These are tokens which are used to represent the structure of the query, inherent to the formal target language of choice, such as SQL-specific tokens like `SELECT` and `WHERE`. Since these tokens have a fixed, example-independent meaning, they can be represented by their respective embedding vectors shared across all examples. Thus, the tokens from $\mathcal{V}^{\mathrm{SQL}}$ are embedded based on a learnable, randomly initialized embedding matrix W^{SQL} which is reused for all examples.

Column id tokens: These tokens are used to refer to specific columns in the table that the question is being asked against.

Column names may consist of several words, which are first embedded and then fed into a single-layer LSTM. The

final hidden state of the LSTM is taken as the embedding vector representing the column. This approach for computing column representations is similar to other work that encode external information to get better representations for rare words (Bahdanau et al. 2017; Ling et al. 2015; Hill et al. 2016).

Input words: To represent input words in the decoder we reuse the vectors from the embedding matrix W^E, which is also used for encoding the question.

Output Layer of the Decoder

The output layer of the decoder takes the current context $\hat{\mathbf{h}}_t$ and the hidden state $\mathbf{y}_t$ of the decoder's LSTM and produces probabilities over the output vocabulary $\mathcal{V}^D$. Probabilities over SQL tokens and column id tokens are calculated based on a dedicated linear transformation, as opposed to the probabilities over input words which rely on a pointer mechanism that enables copying from the input question.

Generating scores for SQL tokens and column id tokens For the SQL tokens ($\mathcal{V}^{\text{SQL}}$), the output scores are computed by the linear transformation: $\mathbf{o}^{\text{SQL}} = U^{\text{SQL}} \cdot [\mathbf{y}_t, \hat{\mathbf{h}}_t]$, where $U^{\text{SQL}} \in \mathbb{R}^{|\mathcal{V}^{\text{SQL}}| \times d^{out}}$ is a trainable matrix. For the *column id tokens* ($\mathcal{V}^{\text{COL}}$), we compute the output scores based on a transformation matrix U^{COL}, holding dynamically computed encodings of all column ids present in the table of the current example. For every column id token, we encode the corresponding column name using an LSTM, taking its final state as a (preliminary) column name encoding $\mathbf{u}^*$, similarly to what is done in the embedding function. By using skip connections we compute the average of the word embeddings of the tokens in the column name, $\mathbf{c}_i$ for $i = 1, \ldots, K$, and add them to the preliminary column name encoding $\mathbf{u}^*$ to obtain the final encoding for the column id:

$$\mathbf{u} = \mathbf{u}^* + \begin{bmatrix} \mathbf{0} \\ \frac{1}{K} \sum_i^K \mathbf{c}_i \end{bmatrix} \quad , \tag{4}$$

where we pad the word embeddings with zeros to match the dimensions of the encoding vector before adding.

The output scores for all column id tokens are then computed by the linear transformation[2]: $\mathbf{o}^{\text{COL}} = U^{\text{COL}} \cdot [\mathbf{y}_t, \hat{\mathbf{h}}_t]$.

Pointer-based copying from the input To enable our model to copy tokens from the input question, we follow a pointer-based (Gu et al. 2016; See et al. 2017) approach to compute output scores over the words from the question. We explore two different copying mechanism, a *shared softmax* approach inspired Gu et al. (2016) and a *point-or-generate* method similar to See et al. (2017). The two copying mechanisms are described in the following.

Point-or-generate: First, the concatenated output scores for SQL and column id tokens are turned into probabilities using a softmax

$$p^{\text{GEN}}(S_t | s_{t-1}, \ldots, s_0, Q) = \mathsf{softmax}([\mathbf{o}^{\text{SQL}}; \mathbf{o}^{\text{COL}}]) \ . \tag{5}$$

Then we obtain the probabilities over the input vocabulary $\mathcal{V}^E$ based on the attention probabilities $\alpha_i^{(t)}$ (Eq. 2) over the question sequence $Q = [q_0, \ldots, q_i, \ldots, q_N]$. To obtain the pointer probability for a token q in the question sequence we sum over the attention probabilities corresponding to all the positions of Q where q occurs, that is

$$p^{\text{PTR}}(q | s_{t-1}, \ldots, s_0, Q) = \sum_{i:q_i=q} \alpha_i^{(t)} \quad . \tag{6}$$

The pointer probabilities for all input tokens $q \in \mathcal{V}^E$ that do not occur in the question Q are set to 0.

Finally, the two distributions p^{GEN} and p^{PTR} are combined into a mixture distribution:

$$\begin{aligned} p(S_t | s_{t-1}, \ldots, s_0, Q) = \gamma p^{\text{PTR}}(S_t | s_{t-1}, \ldots, s_0, Q) \\ + (1-\gamma) p^{\text{GEN}}(S_t | s_{t-1}, \ldots, s_0, Q) \ , \end{aligned} \tag{7}$$

where the scalar mixture weight $\gamma \in [0, 1]$ is given by the output of a two-layer feed-forward neural network, that gets $[\mathbf{y}_t, \hat{\mathbf{h}}_t]$ as input.

Shared softmax: In this approach, we re-use the attention scores $a_i^{(t)}$ (Eq. 1) and obtain the output scores $\mathbf{o}^{\text{E}}$ over the tokens $q \in \mathcal{V}^E$ from the question as follows: for every token q that occurs in the question sequence Q the output score is given by the maximum attention score over all positions in $Q = [q_0, \ldots, q_i, \ldots, q_N]$ where q occurs, i.e. it is given by:

$$\max_{i:q_i=q} a_i \ , \tag{8}$$

while the scores for all input tokens $q \in \mathcal{V}^E$ that do not occur in the question Q are set to $-\infty$. The final output probabilities are then computed based on a single softmax function that takes the output scores of the whole output vocabulary as input:

$$p(S_t | s_{t-1}, \ldots, s_0, Q) = \mathsf{softmax}([\mathbf{o}^{\text{SQL}}; \mathbf{o}^{\text{COL}}; \mathbf{o}^{\text{E}}]) \ . \tag{9}$$

Pretrained Embeddings and Rare Words

We initialize all NL embedding matrices[3] using GloVe embeddings for words covered by GloVe (Pennington et al. 2014) and use randomly initialized vectors for the remaining words. Whereas randomly initialized word embeddings are trained together with the remaining model parameters, we keep GloVe embeddings fixed, since finetuning them led to worse results in our experiments.

We also replace rare words that do not occur in GloVe with a rare word representation in all embedding matrices.

[2] Note that the skip connections in both the question encoder and the column name encoder use padding such that the word embeddings are added to the same regions of $[\mathbf{y}_t, \hat{\mathbf{h}}_t]$ and $\mathbf{u}$, respectively, and thus are directly matched.

[3] W^E simultaneously used for question word embedding in the encoder and input word embedding in the embedding function of the decoder, the embedding matrix W^{CT} for words occurring in column names used in the embedding function of the decoder, and its analogue in the output function.

Coherence of decoded logical forms

The output sequences produced by a unconstrained decoder can be syntactically incorrect and result in execution errors or they can make mistakes against table semantics. We avoid such mistakes by implementing a constrained decoder that exploits task-specific syntactic and semantic rules[4].

The grammar behind the produced sequences is simple and the constraints can be implemented easily by keeping track of the previous token and whether we are in the `SELECT` or `WHERE` clause. In our example discussed earlier (see Figure 1), after a `COND` token, only column id tokens (`L1_Cache`, `FSB_Speed`, ...) can follow, and after a column id token, only an operator token (`OP1`, `OP2`, ...) is allowed if we are currently decoding the `WHERE` clause.

In addition to such syntactic rules, we take into account the types of columns to restrict the set of aggregators and operators that can follow a column id. In the case of WIKISQL, there are two column types: `text` and `float`. Aggregators like average and operators like greater_than only apply on `float`-typed columns and thus are not allowed after `text` columns. We also enforce span consistency when copying tokens, leaving only the choice of copying the next token from the input or terminating copying, if the previous action was a copy action.

Training

We train our models by maximizing the likelihood of a correct logical form given the natural language question. We experiment with teacher forcing (TF) and a non-deterministic oracle (Goldberg and Nivre 2012).

Teacher forcing takes the original linearizations of the query trees (as provided in the dataset) and uses it both for supervision and as input to the decoder. However, in the presence of multiple correct decoding paths, teacher forcing can suffer from suboptimal supervision order, as pointed out by previous work on WIKISQL (Zhong et al. 2017; Xu et al. 2017) and concurrently explored by (Shi et al. 2018).

Non-deterministic Oracle

Instead of forcing the model to follow the original decoding sequence, a non-deterministic oracle enables the exploration of alternative linearizations of the query tree and is an adaptation of Goldberg and Nivre's (2012) algorithm for a *dynamic oracle with spurious ambiguity* (developed in the context of dependency parsing). It is formally described in Algorithm 1, which is invoked at every decoding step t to get a token g_t (used for supervision) and a token x_{t+1} (used as input to the decoder in the next time step). Essentially, the algorithm always picks the best-scored *correct* token for supervision and uniformly samples one of the correct tokens to be used as decoder input in the next time step, if the *overall* best-scored token (over the whole output vocabulary) does not belong to the correct ones. Thus, the oracle explores alternative paths if the decoder would make a mistake in free-running mode.

[4]We use these constraints during prediction only.

Algorithm 1 Non-deterministic oracle

function GETNEXTANDGOLD(p_t, t, $x_{\leq t}$)
  $\mathsf{VNT}_t \leftarrow \texttt{get\_valid\_next}(t, x_{\leq t})$
  $x_{t+1} \leftarrow \text{argmax}_{\mathcal{V}^D}\, p_t$
  $g_t \leftarrow \text{argmax}_{\mathsf{VNT}_t}\, p_t$
  if $x_{t+1} \notin \mathsf{VNT}_t$ **then**
    $x_{t+1} \leftarrow \texttt{random}(\mathsf{VNT}_t)$
  return g_t, x_{t+1}

In the algorithm, p_t is the decoder's output distribution over $\mathcal{V}^D$ at time step t. The set of valid next tokens $\mathsf{VNT}_t \subset \mathcal{V}^D$, from which the correct tree can be reached, is returned by the function `get_valid_next`$(\cdot)$. The query tree can have nodes with either ordered or unordered children (for example, children of the `WHERE` clause are unordered). If we are currently decoding the children of a node with unordered children, all the children that have not been decoded yet are returned as VNT_t. In other cases, VNT_t contains the next token according to the original sequence order.

REINFORCE

The presented oracle is similar to REINFORCE in that it explores alternative paths to generate the same query. In contrast to the oracle, REINFORCE samples the next token (x_{t+1}) according to the predictive distribution p_t and then uses the sampled sequence to compute gradients for policy parameters:

$$\nabla J = \mathbb{E}[\nabla \log(p_t(x_{t+1}))A_t] \tag{10}$$

In Alg. 2, we adapt the oracle into an algorithm equivalent to basic REINFORCE with episode reward A_t set to $+1$ if the sampled sequence produces a correct query and 0 otherwise.

Algorithm 2 Our REINFORCE

function GETNEXTANDGOLD(p_t, t, $x_{\leq t}$)
  $\mathsf{VNT}_t \leftarrow \texttt{get\_valid\_next}(t, x_{\leq t})$
  $x_{t+1} \sim p_t;\ \ x_{t+1} \in \mathsf{VNT}_t$
  $g_t \leftarrow x_{t+1}$
  return g_t, x_{t+1}

Evaluation

To evaluate our approach, we obtain the WIKISQL (Zhong et al. 2017) dataset by following the instructions on the WIKISQL website[5]. The dataset contains a total of 80654 examples. Each example provides a NL question, its SQL equivalent and the table against which the SQL query should be executed. The original training/dev/test splits of WIKISQL use disjoint sets of tables with different schemas.

Experimental Setup

Evaluation: Similarly to previous work, we report (1) sequence match accuracy (Acc_{LF}), (2) query match accuracy

[5]`http://github.com/salesforce/WikiSQL`

(Acc_{QM}) and (3) query execution accuracy (Acc_{EX}). Note that while Acc_{LF} accepts only the original linearizations of the trees, Acc_{QM} and Acc_{EX} accept all orderings leading to the same query.

Training details: After a hyperparameter search, we obtained the best results by using two layers both in the encoder and decoder LSTMs, with every layer of size 600, and embedding size of 300, and applying time-shared dropouts on the inputs of the recurrent layers (dropout rate 0.2) and recurrent connections (dropout rate 0.1). We trained using Adam, with a learning rate of 0.001 and a batch size of 100, a maximum of 50 epochs and early stopping. We also use label smoothing with a mixture weight $\epsilon = 0.2$, as described in Szegedy et al. (2016).

We ran all reported experiments at least three times and report the average of the computed metrics. While the variance of the metrics varies between settings, it generally stays between 0.1 and 0.25 percent for Acc_{QM}.

Results

We present our results, compared to previous and concurrent work in Table 1. Our method compares well against previous work, achieving performance similar to Coarse2Fine (Dong and Lapata 2018) and close to MQAN (McCann et al. 2018) which have more complicated architectures. Approaches using execution-guided decoding (EG) show better performance at the expense of access to table content and repeated querying during decoding, and relies on the assumption that the query should not return empty result sets. The concurrently developed oracle-based[6] approach of Shi et al. (2018) improves upon our investigation of the oracle using the `ANYCOL` technique (see Related Work section).

In the following sections, we provide an ablation study, an in-depth analysis of the influence of the linearization order of query trees, as well as an error analysis. The analysis reveals that the overall improvement in accuracy obtained from using the oracle can be attributed to improved prediction accuracy of WHERE clauses, which contain unordered elements.

Ablation study Starting from the best variant of our model (i.e. the *shared softmax* pointer-generator) and standard TF based training, we want to investigate the role of different model components and the different training approaches.

Table 2 presents the results of this ablation study. Without constraints enforcing the coherence of the decoded logical rule at test time, the results drop by 1.6% Acc_{QM} on the test set. While also using the constraints during training doesn't deteriorate results much, it results in slower training.

Label smoothing (Szegedy et al. 2016) has a significant impact on performance. Label smoothing relaxes the target distribution and thus helps to reduce overfitting. While label smoothing improves the performance of both versions of pointer-generators, it improves the *shared softmax* version by 2% test Acc_{QM}, as opposed to a slightly lower improvement of 1.4% for *point-or-generate*.

Incorporating skip connections into the encoder and decoder of our model improved performance by 0.5% Acc_{QM} on the test set.

Effect of ordering in supervision To investigate the influence of the ordering in the linearizations of queries, we trained our model with teacher forcing and experimented with (1) reversing the original order of conditions in the WHERE clause and (2) training with target sequences where we assigned a different random order to the conditions in every trial. The results indicate that the order of conditions in the linearization matters for the performance of TF based training to a certain degree. Training with a randomly reassigned order of conditions in the WHERE clause results in a 2.5% drop in query accuracy (Acc_{QM}) on the test set. However, reversing the order of conditions does not affect the results.

Furthermore, we trained our model with REINFORCE as well as with the non-deterministic oracle. In both methods, the originally provided order of the target sequence does not matter. Using REINFORCE (indicated by "RL" in Table 3) results in a 1.5% drop in Acc_{QM} on the test set. The oracle as described in Alg. 1 results in an improvement of 0.6% query accuracy on the test set. We can also see that Acc_{LF} for the oracle is significantly lower compared to TF while Acc_{QM} is on par with TF. Given that Acc_{LF} is sensitive to the order of arbitrarily ordered clauses and Acc_{QM} is not, this means that the oracle-trained models effectively learned to use alternative paths.

Comparing the oracle to TF with arbitrarily reordered conditions in the WHERE clause shows that training with TF can suffer from supervision sequences that are not consistently ordered. When training with the oracle, the order of unordered nodes as provided in supervision sequences does not matter. Thus, it can be beneficial (in this case by 3% query accuracy) to use the oracle if the original linearization is arbitrary and can not be made consistent.

Error analysis Table 4 shows accuracies of different parts of the query over the development set of WIKISQL. The main cause of a wrongly predicted SELECT clause is an error in the predicted aggregator, while the main cause of error overall is the prediction of the WHERE clause.

Comparison of errors of models trained with TF versus oracle reveals that oracle-trained models make fewer mistakes in the WHERE clause, showing a 1% improvement (84.4% from 83.4%) in WHERE clause accuracy, which is translated to the 0.8% (73.4% from 72.6%) improvement in full query accuracy (Acc_{QM}) on the validation set.

We find no difference between the accuracies for the SELECT clause between TF and oracle training settings. In both cases, 68.7% of examples with wrongly predicted SELECT clauses had an error in the predicted aggregator, and 36.5% had a wrongly selected column.

Related Work

Earlier work on semantic parsing relied on CCG and other grammars (Zettlemoyer and Collins 2007; Berant et al.

[6]We also investigated non-deterministic oracles in the preprint of this work from May 2018 (`https://openreview.net/forum?id=HJMoLws2z`).

	Dev Accuracies (%)			Test Accuracies (%)		
	Acc_{LF}	Acc_{QM}	Acc_{EX}	Acc_{LF}	Acc_{QM}	Acc_{EX}
Seq2SQL (no RL) (Zhong et al. 2017)	48.2	–	58.1	47.4	–	57.1
Seq2SQL (RL) (Zhong et al. 2017)	49.5	–	60.8	48.3	–	59.4
Pointer-SQL (Wang et al. 2018a)	59.6	–	65.2	59.5	–	65.1
SQLNet (Xu et al. 2017)	–	63.2	69.8	–	61.3	68.0
PT-MAML (Huang et al. 2018)*	63.1	–	68.3	62.8	–	68.0
TypeSQL (Yu et al. 2018)*	–	68.0	74.5	–	66.7	73.5
STAMP (Sun et al. 2018)*	61.7	–	75.1	61.0	–	74.6
Coarse2Fine (Dong and Lapata 2018)	–	–	–	–	71.7	78.5
MQAN (McCann et al. 2018)	–	–	–	72.4	–	80.4
(ours)						
PtrGen-SQL (*shared softmax*)	70.2	72.6	79.0	69.9	72.1	78.4
PtrGen-SQL (*point-or-generate*)	70.0	72.4	78.5	69.7	71.7	78.0
PtrGen-SQL (*shared softmax*) + oracle	56.2	73.4	79.4	55.0	72.7	78.8
(EG-based or concurrent work)						
Pointer-SQL + EG(5) (Wang et al. 2018b)	67.5	–	78.4	67.9	–	78.3
Coarse2Fine + EG(5) (Wang et al. 2018b)	76.0	–	84.0	75.4	–	83.8
IncSQL + oracle + `ANYCOL` (Shi et al. 2018)	49.9	–	84.0	49.9	–	83.7
IncSQL + oracle + `ANYCOL` + EG(5) (Shi et al. 2018)	51.3	–	87.2	51.1	–	87.1

Table 1: Evaluation results for our approach (middle section) and comparison with previously reported results (top part) and concurrent work or EG-based systems (bottom part). Entries marked by * are trained and evaluated using a slightly different version of the WikiSQL dataset. Some values in the table, indicated by "–", could not be filled because the authors did not report the metric or the metric was not applicable.

	Dev Accs (%)		Test Accs (%)	
	Acc_{LF}	Acc_{QM}	Acc_{LF}	Acc_{QM}
PtrGen-SQL (*shared softmax*)	70.2	72.6	69.9	72.1
· no constraints	68.6	70.9	68.6	70.5
· using constraints during training	69.8	72.2	69.8	71.9
· no label smoothing	68.3	70.5	68.4	70.1
· no label smoothing (*point-or-generate*)	68.7	70.7	68.5	70.3
· no skip connections	69.6	72.0	69.4	71.6

Table 2: Performance of different variations of our approach.

2013). With the recent advances in recurrent neural networks and attention (Bahdanau et al. 2014; See et al. 2017), neural translation based approaches for semantic parsing have been developed (Dong and Lapata 2016; Liang et al. 2016; Rabinovich et al. 2017).

Labels provided for supervision in semantic parsing datasets can be given either as execution results or as an executable program (logical form). Training semantic parsers on logical forms yields better results than having only the execution results (Yih et al. 2016) but requires a more elaborate data collection scheme. Significant research effort has been dedicated to train semantic parsers only with execution results. Using policy gradient methods (such as REINFORCE) is a common strategy (Liang et al. 2016; Zhong et al. 2017). Alternative methods (Krishnamurthy et al. 2017; Iyyer et al. 2017; Guu et al. 2017) exist, which also maximize the likelihood of the execution results.

Related to the ordering issue, the work of Vinyals et al. (2016) also investigates the effect of ordering in the linearization of target structures with unordered elements. We adapt the approach of Goldberg and Nivre (2012) that was developed in the context of dependency parsing. Goldberg and Nivre (2012) experiment with two versions of the dynamic oracle, one that handles spurious ambiguity, and one that is also able to recover from incorrect actions after which the gold tree can not be reached.

Similar to the WIKISQL dataset that we used in our experiments are the ATIS (Dahl et al. 1994) and WIKITABLEQUESTIONS (Pasupat and Liang 2015) datasets, which also focus on question answering over tables. In contrast to WIKISQL however, both ATIS and WIKITABLEQUESTIONS are significantly smaller and the latter does not provide logical forms for supervision and thus requires training with execution results as supervision (Neelakantan et al. 2016; Haug et al. 2018; Krishnamurthy et al. 2017). SQA (Iyyer et al. 2017) is a dataset derived from WIKITABLEQUESTIONS and focuses on question answering in a dialogue context.

Previous work on WIKISQL (Zhong et al. 2017; Xu et

al. 2017; Huang et al. 2018; Yu et al. 2018; Wang et al. 2018a) generally incorporate both slot-filling and sequence decoding, predicting the SELECT clause arguments with separate slot-filling networks, and also include some form of a pointing mechanism. Seq2SQL (Zhong et al. 2017) proposes an *augmented pointer network* that also uses a pointer but encodes the question, column names and SQL tokens together, and completely relies on a pointer to generate the target sequence. To avoid the *order-matters* problem, SQLNet (Xu et al. 2017) proposes a sequence-to-set model that makes a set inclusion prediction in order to avoid decoding the conditions in any particular order. Both predict the SELECT clause arguments using separate specialized predictors. Zhong et al. (2017) also use Dong and Lapata (2016)'s SEQ2SEQ model as a baseline, however, get poor performance due to the lack of pointer and column encoders. Yu et al. (2018) build upon SQLNet (Xu et al. 2017)'s slot filling approach, proposing several improvements such as weight sharing between SQLNet's subnetworks, and incorporate precomputed type information for question tokens in order to obtain a better question encoding. Wang et al. (2018a) develop a model similar to ours; they propose a SEQ2SEQ model with copy actions. Similarly to Zhong et al. (2017), they encode the concatenation of column names and the question. Similarly to our work, they use a constrained decoder to generate SQL tokens or copy column names or question words from the encoded input sequence. In contrast to Wang et al. (2018a), we encode column names separately, and independently from the question. Huang et al. (2018) experiment with meta-learning (MAML), using Wang et al. (2018a)'s model. STAMP (Sun et al. 2018) presents a "multi-channel" decoder that considers three types of tokens (SQL, column, cell), and mixes the distributions for each type using coefficients produced by a trainable subnetwork. Compared to STAMP, we do not encode cells (we assume no knowledge of table contents) and instead use a pointer to copy values of conditions from the input. Coarse2Fine (Dong and Lapata 2018) explores a middle ground between purely sequence and tree decoding models (Alvarez-Melis and Jaakkola 2016; Dong and Lapata 2016) and proposes a two-stage decoding process, where first a template (sketch) of the query is decoded and subsequently filled in.

Very recent and concurrent work on WIKISQL explores execution-guided (EG) decoding (Wang et al. 2018b) and non-deterministic oracles (Shi et al. 2018). Execution-guided decoding keeps a beam of partially decoded queries, which are filtered based on the execution results, that is, a partially encoded query is not taken further into account if it can not be parsed, produces a runtime error, or returns no results after execution. This requires multiple queries to be executed against the database while decoding. In our work, we try to avoid parsing and semantics-related runtime errors more efficiently by using decoding constraints. We suspect that a significant part of the improvement due to EG in decoding relies on the assumption that execution results should not be empty. However, we believe this assumption does not hold in general, due to the existence of queries for which an empty set is the correct answer. IncSQL (Shi et al. 2018) also uses EG decoding, as well as a non-deterministic oracle extended with the `ANYCOL` token, which adds the option to produce a wildcard column token that matches any column. During training, the wildcard column token is provided as an alternative to the true column token in the supervision sequence if it can be unambiguously resolved to the true column using the condition value. IncSQL's model goes beyond ours by adding self- and cross-serial attention and a final inter-column BiLSTM encoder. They also feed column attention and question attention summaries as an input to the decoder.

	Dev Accs (%)		Test Accs (%)	
	Acc_{LF}	Acc_{QM}	Acc_{LF}	Acc_{QM}
Original order (TF)	70.2	72.6	69.9	72.1
· Reversed (TF)	–	72.6	–	72.1
· Arbitrary (TF)	–	70.4	–	69.6
· RL	59.9	71.4	59.1	70.6
· Oracle	56.2	73.4	55.0	72.7

Table 3: Results for different target tree linearizations.

	TF	oracle
Whole Query	72.6	73.4
· SELECT	85.5	85.5
· Aggregator	90.0	90.0
· Column	94.7	94.7
· WHERE	83.4	84.4

Table 4: Error Analysis: Acc_{QM} of different query parts on the development set for TF and oracle-trained *shared softmax* models.

Conclusion

In this work we present a SEQ2SEQ model adapted to the semantic parsing task of translating natural language questions to queries over tabular data. We investigated how the ordering of supervision sequences during training affects performance, concluding that the order of conditions in the linearization of the query tree matters to a certain degree for WIKISQL. In this context, we also evaluated the use of REINFORCE and a non-deterministic oracle for training the neural network-based semantic parser. Our experiments revealed that REINFORCE does not improve results and the oracle provides a small improvement, which can be attributed to improved decoding of the WHERE clause. Furthermore, from the results we can conclude that training with a non-deterministic oracle is advisable if the original linearizations are inconsistently ordered.

Acknowledgement

We acknowledge the support of the European Union H2020 framework ITN projects WDAqua (grant no. 642795) and Cleopatra (grant no. 812997).

References

Alvarez-Melis, D., and Jaakkola, T. S. 2016. Tree-structured decoding with doubly-recurrent neural networks.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2014*.

Bahdanau, D.; Bosc, T.; Jastrzebski, S.; Grefenstette, E.; Vincent, P.; and Bengio, Y. 2017. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286*.

Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP 2013*, 1533–1544.

Dahl, D. A.; Bates, M.; Brown, M.; Fisher, W.; Hunicke-Smith, K.; Pallett, D.; Pao, C.; Rudnicky, A.; and Shriberg, E. 1994. Expanding the Scope of the ATIS Task: The ATIS-3 Corpus. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, 43–48. Stroudsburg, PA, USA: Association for Computational Linguistics.

Dong, L., and Lapata, M. 2016. Language to logical form with neural attention. In *Proceedings of ACL 2016*.

Dong, L., and Lapata, M. 2018. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*.

Goldberg, Y., and Nivre, J. 2012. A dynamic oracle for arc-eager dependency parsing. *Proceedings of COLING 2012*.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Guu, K.; Pasupat, P.; Liu, E.; and Liang, P. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of ACL 2017*.

Haug, T.; Ganea, O.-E.; and Grnarova, P. 2018. Neural multi-step reasoning for question answering on semi-structured tables. 611–617.

Hill, F.; Cho, K.; Korhonen, A.; and Bengio, Y. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association of Computational Linguistics*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Comput.*

Huang, P.-S.; Wang, C.; Singh, R.; Yih, W.-t.; and He, X. 2018. Natural language to structured query generation via meta-learning. *arXiv preprint arXiv:1803.02400*.

Iyyer, M.; Yih, W.-t.; and Chang, M.-W. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of ACL 2017*, volume 1, 1821–1831.

Krishnamurthy, J.; Dasigi, P.; and Gardner, M. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of EMNLP 2017*.

Liang, C.; Berant, J.; Le, Q.; Forbus, K. D.; and Lao, N. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Ling, W.; Luís, T.; Marujo, L.; Astudillo, R. F.; Amir, S.; Dyer, C.; Black, A. W.; and Trancoso, I. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of EMNLP 2015*.

McCann, B.; Keskar, N. S.; Xiong, C.; and Socher, R. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Neelakantan, A.; Le, Q. V.; and Sutskever, I. 2016. Neural programmer: Inducing latent programs with gradient descent. In *ICLR 2016*.

Pasupat, P., and Liang, P. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*, 1532–1543.

Rabinovich, M.; Stern, M.; and Klein, D. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of ACL 2017*, volume 1, 1139–1149.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Shi, T.; Tatwawadi, K.; Chakrabarti, K.; Mao, Y.; Polozov, O.; and Chen, W. 2018. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.

Sun, Y.; Tang, D.; Duan, N.; Ji, J.; Cao, G.; Feng, X.; Qin, B.; Liu, T.; and Zhou, M. 2018. Semantic parsing with syntax- and table-aware sql generation. In *Proceedings of ACL 2018*.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of CVPR 2016*, 2818–2826.

Vinyals, O.; Bengio, S.; and Kudlur, M. 2016. Order matters: Sequence to sequence for sets.

Wang, C.; Brockschmidt, M.; and Singh, R. 2018a. Pointing out sql queries from text.

Wang, C.; Huang, P.-S.; Polozov, A.; Brockschmidt, M.; and Singh, R. 2018b. Execution-guided neural program decoding. *arXiv preprint arXiv:1807.03100*.

Xu, X.; Liu, C.; and Song, D. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Yih, W.-t.; Richardson, M.; Meek, C.; Chang, M.-W.; and Suh, J. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of ACL 2016*, volume 2, 201–206.

Yu, T.; Li, Z.; Zhang, Z.; Zhang, R.; and Radev, D. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. *arXiv preprint arXiv:1804.09769*.

Zettlemoyer, L., and Collins, M. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of EMNLP-CoNLL 2007*.

Zhong, V.; Xiong, C.; and Socher, R. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

TallyQA: Answering Complex Counting Questions

Manoj Acharya, Kushal Kafle, Christopher Kanan
Chester F. Carlson Center for Imaging Science
Rochester Institute of Technology
{ma7583, kk6055, kanan}@rit.edu

Abstract

Most counting questions in visual question answering (VQA) datasets are simple and require no more than object detection. Here, we study algorithms for complex counting questions that involve relationships between objects, attribute identification, reasoning, and more. To do this, we created TallyQA, the world's largest dataset for open-ended counting. We propose a new algorithm for counting that uses relation networks with region proposals. Our method lets relation networks be efficiently used with high-resolution imagery. It yields state-of-the-art results compared to baseline and recent systems on both TallyQA and the HowMany-QA benchmark.

Introduction

Open-ended counting systems take in a counting question and an image to predict a whole number that answers the question. While object recognition systems now rival humans (He et al. 2016), today's best open-ended counting systems perform poorly (Kafle and Kanan 2017b; Chattopadhyay et al. 2017). This could be due to an inability to detect the correct objects or due to an inability to reason about them. To address this, we distinguish between *simple* and *complex* counting questions (see Fig. 1). Simple counting questions only require object detection, e.g., "How many dogs are there?" Complex questions require deeper analysis, e.g., "How many dogs are eating?"

Open-ended counting is a special case of visual question answering (VQA) (Antol et al. 2015; Malinowski and Fritz 2014), in which the goal is to answer open-ended questions about images. The best VQA systems pose it as a classification problem where the answer is predicted from convolutional visual features and the question (Kafle and Kanan 2017a). While this succeeds for many question types, it works poorly for counting (Kafle and Kanan 2017b; Chattopadhyay et al. 2017). Recently, better results were achieved by using region proposals generated by object detection algorithms (Trott, Xiong, and Socher 2018; Zhang, Hare, and Prügel-Bennett 2018). However, datasets mostly contain simple counting questions, as shown in Table 1. Due to their rarity, complex questions need to be analyzed separately to determine if a model is capable of answering them.

This paper makes three major contributions:

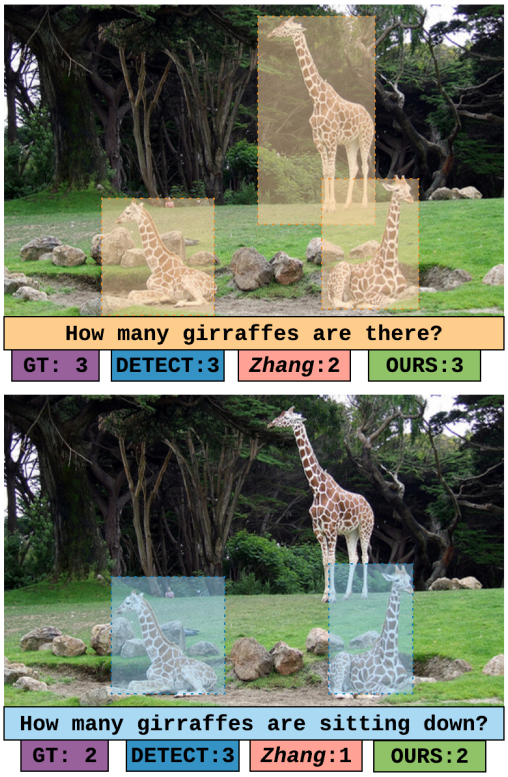


Figure 1: Counting datasets consist mostly of simple questions (top) that can be answered solely using object detection. We study complex counting questions (bottom) that require more than object detection using our new TallyQA dataset.

1. We describe TallyQA, the largest open-ended counting dataset. TallyQA is designed to study both simple questions that require only object detection and complex questions that demand more. It is now publicly available.
2. We propose the relational counting network (RCN), a new algorithm for counting that infers relationships between objects and background image regions. It is inspired by relation networks, with modifications to handle a dynamic number of image regions and to explicitly incorporate background information.
3. We show that RCN surpasses state-of-the-art methods for

Reprint from the AAAI-19 proceedings

open-ended counting on both TallyQA and the HowMany-QA benchmark.

Related Work

VQA Datasets & Counting

Popular VQA datasets contain a significant number of counting questions, e.g., about 7% in COCO-QA (Ren, Kiros, and Zemel 2015), 10% of VQA1 (Antol et al. 2015), 10% of VQA2 (Goyal et al. 2017), and 10% of TDIUC (Kafle and Kanan 2017b). There are also counting specific VQA datasets. CountQA (Chattopadhyay et al. 2017) was created by importing question-answer (QA) pairs from the validation split of VQA1 and COCO-QA. This small dataset has only 2,287 test QA pairs. Recently, HowMany-QA (Trott, Xiong, and Socher 2018) was created by importing QA pairs from Visual Genome and VQA2, and it is considered the gold standard for open-ended counting. As shown in Table 1, complex questions are scarce in these datasets. Simple questions can be solved using solely an object detection algorithm, so they do not appropriately test a system's ability to answer arbitrary counting questions, including those requiring reasoning or attribute recognition. Our new dataset, TallyQA, is designed to evaluate both simple and complex counting questions, enabling these and other capabilities to be appropriately evaluated. Note that we have limited our discussion to natural language VQA systems which pose different challenges compared to VQA on synthetic datasets, which are often designed for specific purposes (Johnson et al. 2017; Kafle et al. 2018; Zhang et al. 2016).

Algorithms for Open-Ended Counting

Open-ended counting systems take as input a "How many ...?" question and an image and then output a count. This is a VQA sub-problem. For counting, there are two general approaches. The first involves inferring the count directly in an end-to-end framework operating on high-level CNN features. The second approach is to detect object bounding boxes or region proposals, and then aggregate question relevant bounding boxes. While there are many direct methods, over the past year region-based schemes for open-ended counting have been studied.

Direct Methods. State-of-the-art VQA systems train a classifier to predict the answer from the image and question. Typically, image features are encoded using a CNN that was pre-trained on ImageNet, and questions are encoded using a recurrent neural network (RNN). Many innovations involve different ways of combining image and question features (Lu et al. 2016; Fukui et al. 2016; Ben-younes et al. 2017; Kafle and Kanan 2016), modular networks (Andreas et al. 2016), data-augmentation (Kafle, Yousefhussien, and Kanan 2017) among many others.

Direct methods perform poorly at counting for *real-world* image datasets. In Kafle and Kanan (2017b), three state-of-the-art VQA algorithms were compared to baselines on TDIUC's counting questions. The best performing method, MCB, achieved 51% accuracy, which was only 6% better than an image-blind (question-only) model. This was true even though most of TDIUC's counting questions are simple. This suggests these methods are primarily exploiting scene and language priors. For VQA2, the best method (Teney et al. 2017) of the CVPR-2017 VQA Workshop challenge achieved 69% overall, but only 47% accuracy on number questions, most of which are counting.

	VQA2	TDIUC	HowMany-QA	**TallyQA (Us)**
Simple	78,455	148,719	68,956	211,430
Complex	34,799	16,043	37,400	76,477
Total	113,254	164,762	106,356	287,907

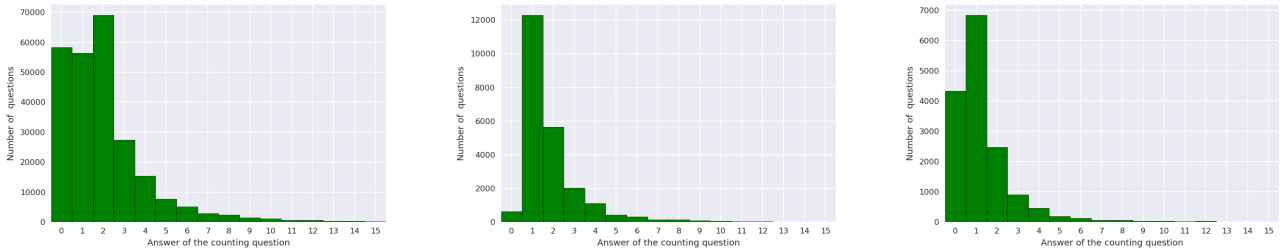
Table 1: The number of counting questions for previous VQA datasets compared to TallyQA dataset.

We hypothesize that the inability of VQA algorithms to count is due to the way their architectures are designed. These systems operate on image embeddings computed using a CNN. Mean pooling and weighted mean pooling (attention) operations may destroy information that can be used to determine how many objects of a particular type are present.

Counting Specific Systems. While counting has long been studied for specific computer vision problems (Zhang et al. 2015; Dalal and Triggs 2005; Wang and Wang 2011; Ryan et al. 2009; Ren and Zemel 2017), only recently has open-ended counting in natural scenes been studied. Chattopadhyay et al. (2017) studied open-ended counting in typical scenes, and they evaluated three counting-specific methods: DETECT, GLANCE, and SUBITIZE. DETECT is built on top of an object detection algorithm, which was Fast R-CNN (Girshick 2015) in their implementation. DETECT works by finding the first noun in a question and then matching it to the closest category the detection algorithm has been trained for (e.g., COCO objects). GLANCE uses a shallow multi-layer perceptron (MLP) to regress for specific object counts from a CNN embedding, with the appropriate output unit chosen based on the first noun. SUBITIZE involves breaking the image into a grid, extracting image embeddings from each grid location, aggregating information across grids using an RNN, and then predicting the count for every class in the dataset. Although none of these methods are capable of handling complex questions, all of them outperformed MCB, which was a state-of-the-art VQA model.

Recently, Trott, Xiong, and Socher (2018) and Zhang, Hare, and Prügel-Bennett (2018) both created algorithms for open-ended counting in natural scenes that are built on top of object proposals generated by an object detection algorithm trained on Visual Genome. Trott *et al*. created the ILRC algorithm, which redefines counting as a sequential object selection problem. ILRC uses reinforcement learning to select the objects that need to be counted based on the question. Zhang *et al*. created a method that uses object detection and then constructs a graph of all detected objects based on how they overlap. Edges in the graph are removed based on several heuristics to ensure that duplicated objects are only counted once.

Both Trott *et al*. and Zhang *et al*. operate on region propos-



(a) Train (b) Test-Simple (c) Test-Complex

Figure 2: Histogram of answer counts for each of the three splits of TallyQA.

als and loosely based on the idea of filtering out irrelevant boxes based on the question, *i.e.* selecting a subset of question relevant region proposals. However, successfully determining which boxes should be counted for a given question often requires comparing it with other object proposals (required for duplicate detection, comparative and positional reasoning, etc.), and the background (for modeling context, finding relative size, etc.). Since neither of these algorithms performs any relational or comparative reasoning between the boxes, they may have an impaired ability to answer complex questions. Here, our RCN model applies relational reasoning to object-object and object-background pairs, giving it a more robust capability to answer complex and relational questions. Indeed, our experiments show that RCN outperforms other models on complex questions.

The TallyQA Dataset

Complex counting questions are rare in existing datasets. This prompted us to create TallyQA. TallyQA's test set is split into two parts: Test-Simple for simple counting questions and Test-Complex for complex counting questions. We gathered new complex questions using Amazon Mechanical Turk (AMT), and imported both simple and complex questions from other datasets. Table 1 shows the total number of questions in TallyQA compared to others, and it has over twice as many complex questions. The number of questions in the train and test sets by source is given in Table 2. Fig. 4 shows example images and questions. Test-Simple and Test-Complex contain images from only Visual Genome, and Train has images from COCO and Visual Genome.

Collecting New Complex Questions

To gather new complex questions, we developed targeted AMT tasks that yielded 19,500 complex questions for 17,545 unique images. These tasks were designed to fight the biases in earlier datasets, where simple counting questions were predominantly asked (Kafle and Kanan 2017a). TallyQA's images are drawn from both COCO and Visual Genome, which provides more variety than COCO alone. About 800 unique annotators provided QA pairs. For all tasks, annotators were not allowed to submit obviously simple questions, e.g., "How many x?" and "How many x in the photo?" We manually checked AMT questions to ensure they were complex, and we removed poor quality questions.

Split	Questions	Images
Train	249,318	132,981
AMT	3,902	3,494
Imported	245,416	129,487
Test-Simple	22,991	18,411
AMT	0	0
Imported	22,991	18,411
Test-Complex	15,598	14,051
AMT	15,598	14,051
Imported	0	0

Table 2: Number of questions and images in TallyQA.

We endeavored to ensure non-zero complex questions were *difficult*, e.g., "How many men are wearing glasses?" is not difficult if all of the men in the image are wearing glasses. To do this, annotators were told to ask questions in which there were counter examples, e.g., to ask "How many men are wearing glasses?" only if it had an answer greater than zero, and the contrary question "How many men are *not* wearing glasses?" had an answer greater than zero.

We created a separate task to generate hard complex questions with zero as the answer. Annotators were asked to make questions about objects with attributes not observed in the image, e.g., asking the question "How many dogs have spots?" when there was a dog *without* spots in the image. Similar examples were shown to annotators before annotation.

Importing Questions from Other Datasets

TallyQA also contains questions imported from VQA2 and Visual Genome. A similar approach was used to create HowMany-QA (Trott, Xiong, and Socher 2018) and TDIUC (Kafle and Kanan 2017b). We imported all questions beginning with the phrase "How many..." with answers that were whole numbers between 0–15. Following Kafle and Kanan (2017b), for VQA2, we required that 5 of the 10 annotators give the same answer. Although these questions were generated by humans, as seen in Table 1, most are simple.

We also imported synthetic counting questions from TDIUC (Kafle and Kanan 2017b). These questions were generated for COCO images using its semantic annotations. The creators used a variety of templates to introduce variation in

the questions and used heuristics to avoid answer ambiguity. All template generated questions from TDIUC are simple. In addition to templates, we used their method for making "absurd" questions to create both simple and complex zero count questions. To do this, we first find the objects absent from an image based on its COCO annotations. Then, we randomly sample the counting questions from the rest of the dataset that ask about counting these objects.

Classifying Simple and Complex Questions

The Test-Complex dataset was made using only new, human vetted complex questions from AMT. Because simple questions are common in existing datasets like VQA2, we used imported questions to make Test-Simple. To do this, we developed a classifier to determine if a question was simple.

Our simple-complex classifier is made from a set of linguistic rules. First, any substrings such as "...in the photo?" or "...in the image?" were removed from the question. Then, we used SpaCy to do part of speech tagging on the remaining substring. It was classified as simple if it had only one noun, no adverbs, and no adjectives, otherwise it was deemed complex. This will classify questions such as "How many dogs?" as simple and "How many brown dogs?" as complex.

Every question classified as simple by our rules will be correct (i.e., the false positive rate is zero), making it suitable for creating Test-Simple, but it may sometimes classify simple questions as complex (i.e., the false negative rate is non-zero). For example, the question "How many men are wearing red hats to the left of the tree?" would be classified as complex by our classifier. However, if there was only a single person in the image then it is not truly a complex question, despite the apparent complexity. These kinds of questions are rare and our simple-complex classifier works robustly, but it is possible that it will underestimate the number of simple questions and overestimate the number of complex when used to characterize a dataset. For this reason, we only use human-vetted questions in TallyQA's Test-Complex set.

Dataset Splits & Statistics

TallyQA is split into one training split (Train) and two test splits: Test-Simple and Test-Complex. Using our simple-complex classifier, Train was found to have 188,439 simple and 60,879 complex questions. The number of questions in each split is given in Table 2. The test splits are comprised exclusively of Visual Genome imagery, and no images in the test splits are used in training.

A New Framework for Complex Counting

Our RCN model, depicted in Fig. 3, is formulated as a modified relation network (RN) (Santoro et al. 2017) that can reason about the nature of relationships between image regions. RCN uses the question Q to guide its processing of a list of n foreground region proposals, $O = \{o_1, o_2, \ldots, o_n\}$, and m background regions, $B = \{b_1, b_2, \ldots, b_m\}$, with $o_i \in \mathbb{R}^K$ and $b_j \in \mathbb{R}^K$. Formally, our RCN model is the combination of two RN sub-networks, i.e.,

$$\text{Count}(O, B, Q) = h_\gamma \left(\text{RN}(O, O) \oplus \text{RN}(O, B) \right), \quad (1)$$

where $\oplus$ denotes concatenation, $RN(O, O)$ represents the RN that infers the relationship between foreground regions, $RN(O, B)$ represents the RN responsible for inferring the relationship between each foreground and background region, and h_γ is a neural network with parameters γ that predicts the final count.

The RN for predicting the relationship between foreground proposals in the context of question Q is given by

$$\text{RN}(O, O) = f_{\phi_1} \left(\sum_{i,j} g_{\theta_1}(o_i, o_j, s_{ij}, Q) \right), \quad (2)$$

where f_{ϕ_1} and g_{θ_1} are neural networks with parameters ϕ_1 and θ_1, respectively, that each output a vector, and the vector s_{ij} encodes spatial information about the i-th and j-th proposals. Like the original RN model, the sum is computed over all n^2 pairwise combinations. Similarly, the RN for predicting the relationship of each proposal to the background is given by,

$$\text{RN}(O, B) = f_{\phi_2} \left(\sum_{i,j} g_{\theta_2}(o_i, b_j, s_{ij}, Q) \right), \quad (3)$$

where f_{ϕ_2} and g_{θ_2} are neural networks with parameters ϕ_2 and θ_2, respectively, that output vectors. RCN has two major innovations over the original RN approach.

The original RN used raw CNN feature map indices as regions. This worked well for CLEVR, but this approach works poorly for real-world VQA datasets that require processing at higher resolutions (e.g., VQA2). RCN overcomes this problem by using region proposals. As input, the original RN model used the d^2 elements in a $d \times d$ convolutional feature map, which were each tagged with their spatial coordinates. This means it computed d^4 pairwise relationships. For recent direct VQA methods, a CNN feature map is typically 14×14, meaning that 38,416 comparisons would be needed per counting query. In contrast, RCN's proposal generator produces only 31.12 foreground regions and 16 background patches per image, so only $31.12^2 + (31.12 \times 16) = 1466$ comparisons are made, on average. By using proposals, RCN reduces the number of comparisons by a factor of 26 and scales to real-world imagery, whereas the original RN model used lower resolution imagery and was only evaluated on CLEVR (Johnson et al. 2017), a synthetic dataset that has simple geometric shapes and a plain background.

RCN's second innovation is the explicit incorporation of the background. For queries such as "How many dogs are laying in the grass?" it is necessary to consider background entities (stuff) that are ignored by object detection systems. RCN uses m image background patches, and computes the relationships of each region with each background patch, enabling the background to be studied with relatively few comparisons. In contrast, the original RN model did not explicitly deal with the background, but it was likely unnecessary due to the simple scenes in CLEVR. Explicitly modeling the background can help answer complex counting questions, which often involve attributes of background objects or relationships between objects and background entities.

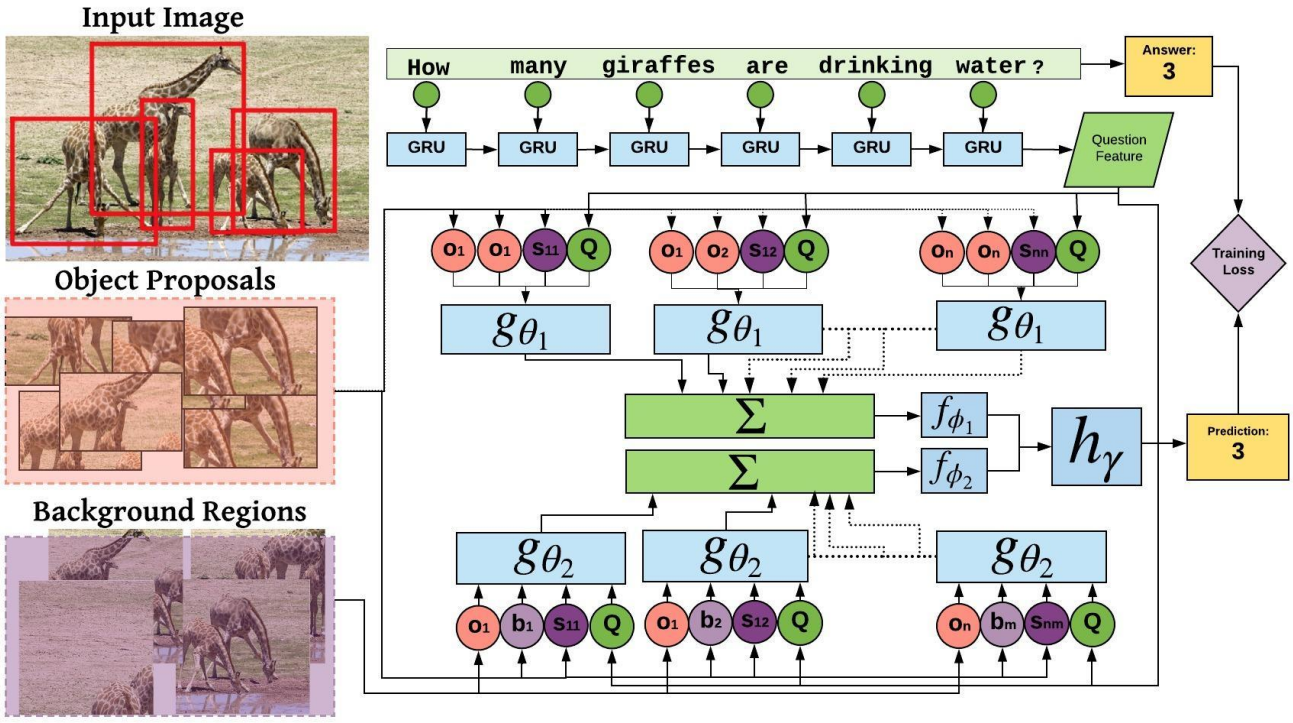


Figure 3: Our RCN model computes the relationship between foreground regions as well as the relationships between the these regions and the background to efficiently answer complex counting questions. In this example, the system needs to look at the relationship of each giraffe to each other and with the water (background).

Internally, RCN uses the spatial relationship between regions o_i and o_j to help predict the count. Using s_{ij} is critical to ensuring each object is counted only once during prediction, and it enables RCN to learn to do non-maximal suppression to cope with overlapping proposals. The spatial relationship vector is given by

$$s_{ij} = \left[\ell_i, \ell_j, \xi_{ij}, IoU_{ij}, \frac{IoU_{ij}}{A_i}, \frac{IoU_{ij}}{A_j}\right], \quad (4)$$

where ℓ_i and ℓ_j encode the spatial information of each proposal individually, A_i and A_j are the area of proposals, ξ_{ij} is the dot product between each proposal's CNN features to model how visually similar they are, and IoU_{ij} is the intersection over union between the two proposals. The vector $\ell_i = \left[\frac{x_{min}}{W}, \frac{y_{min}}{H}, \frac{x_{max}}{W}, \frac{y_{max}}{H}, \frac{x_{max}-x_{min}}{W}, \frac{y_{max}-y_{min}}{H}\right]$, where (x_{min}, x_{max}) and (y_{min}, y_{max}) represent the top-left and bottom-right corners of proposal i, and W and H are the width and height of the image, respectively.

Training & Implementation Details

The question Q is embedded using a one layer GRU that takes as input pre-trained 300-dimensional Glove vectors for each word in the question (Pennington, Socher, and Manning 2014) and outputs a vector of 1024 dimension. The GRU used in all models are regularized using a dropout of 0.3.

For foreground proposals, we use the boxes and CNN features produced by Faster R-CNN (Ren et al. 2015) with ResNet-101 as its backbone. The Faster R-CNN model is trained to predict boxes in Visual Genome, which contain a wide variety of objects and attributes. This approach for generating proposals was pioneered by Anderson et al. (2018), and has since been used by multiple VQA systems.

For the background patches, we extract ResNet-152 features from the entire image before the last pooling layer and then apply average pooling over these features to reduce them to a 4×4 grid. Each of these 2048-dimensional vectors represents a 112×112 pixel background region. In RCN, g_θ has three hidden layers and g_ϕ has one hidden layer, which each consist of 1024 rectified linear units (ReLUs). The outputs of these networks are then concatenated and passed to h_γ, which has one hidden layer with 1024 units and ReLU activation. The softmax output layer treats counting as a classification task, and it is optimized using cross-entropy loss. RCN is trained using the Adam optimizer with a learning rate of $7e^{-4}$ and a batch size of 64 samples.

Experiments

In this section, we describe a series of experiments to evaluate the efficacy of multiple algorithms on both simple and complex counting questions.

	HowMany-QA		TallyQA Test-Simple		TallyQA Test-Complex	
	ACC	RMSE	ACC	RMSE	ACC	RMSE
Guess-1	33.8	3.74	53.5	1.78	43.9	1.57
Guess-2	32.1	3.34	24.5	1.56	15.9	1.69
Q-Only	37.1	3.51	44.6	1.74	39.1	1.75
I-Only	37.3	3.49	46.1	1.71	26.4	1.69
Q+I	40.5	3.17	54.7	1.44	48.8	1.57
DETECT	43.3	3.66	50.6	2.08	15.0	4.52
MUTAN	45.5	2.93	56.5	1.51	49.1	1.59
Zhang *et al.*	54.7	2.59	70.5	1.15	50.9	1.58
IRLC	56.1	2.45	–	–	–	–
RCN (Ours)	**60.3**	**2.35**	**71.8**	**1.13**	**56.2**	**1.43**

Table 3: Performance breakdown on TallyQA and Howmany-QA datasets using accuracy (%) and RMSE.

Models Evaluated

We compare RCN against two state-of-the-art models specifically for open-ended counting: Zhang, Hare, and Prügel-Bennett (2018) and IRLC (Trott, Xiong, and Socher 2018). We also compare against MUTAN (Ben-younes et al. 2017), one of best direct VQA methods. Lastly, we compare RCN to six baseline counting models:

1. **Guess-1**: Answer 1 for all questions.
2. **Guess-2**: Answer 2 for all questions.
3. **Q-Only**: An image-blind MLP model with a hidden layer of 1024 units that uses only the question. The question features are obtained from the last hidden layer of the same RNN architecture used by our RCN model.
4. **I-Only**: A question-blind MLP that has one hidden layer with 1024 units.
5. **Q+I**: An MLP with 1024 hidden units that uses both image and question features.
6. **DETECT**: DETECT is an upgraded version of the method from (Chattopadhyay et al. 2017). The main difference is that we use the more recent YOLOv2 (Redmon and Farhadi 2017) method instead of Fast R-CNN. DETECT extracts the first noun from the question. It then finds the most semantically similar category that YOLOv2 was trained on to that noun based on word similarity, and then it outputs the total number of YOLOv2 boxes produced for that category.

MUTAN, I-Only, and Q+I use ResNet-152 features. Q-Only, I-Only, Q+I, MUTAN, Zhang *et al.* , and RCN all use cross-entropy loss and treat counting as a classification problem. Before evaluation, the output of all models was rounded to the nearest whole number and constrained to be within the range of values in the datasets.

Results

Results for all methods on HowMany-QA and both of TallyQA's test sets are given in Table 3. Following earlier work (Chattopadhyay et al. 2017; Trott, Xiong, and Socher 2018; Zhang, Hare, and Prügel-Bennett 2018), we compute both accuracy and RMSE. RMSE captures that larger errors should be penalized more heavily.

HowMany-QA. HowMany-QA is made by combining counting questions from VQA2 and Visual Genome, so good performance on it serves as a surrogate for good performance on VQA2. HowMany-QA is the best-known dataset for open-ended counting. RCN, IRLC, and Zhang *et al.* all use identical region proposals and CNN features.

RCN obtains the highest accuracy on HowMany-QA, outperforming IRLC, which was the best-known result. Zhang *et al.* achieves the third-highest accuracy. Kim, Jun, and Zhang (2018) used the Zhang *et al.* method to answer VQA2's counting questions. Although they achieved only third best overall in the CVPR 2018 VQA2 Workshop Challenge, they won for number questions.

	Test-Simple		Test-Complex	
	ACC	RMSE	ACC	RMSE
RCN – No Background	69.4	1.18	51.8	1.50
RCN – Full	**71.8**	**1.13**	**56.2**	**1.43**

Table 4: Performance on TallyQA using accuracy (%) and RMSE showing the advantage of using background relationships compared to a version of RCN that omits them.

TallyQA. Example outputs for TallyQA are shown in Fig. 4. IRLC's authors were unable to share code with us, so we could not test IRLC on TallyQA. Zhang *et al.* uses the same Faster R-CNN region proposals and CNN features as RCN.

For Test-Simple, RCN achieves the best accuracy, with Zhang *et al.* performing only slightly worse. On Test-Complex, RCN also achieves the highest accuracy. The next best method is again Zhang *et al.* , but there is a greater gap between the two models. This may be because Zhang *et al.* does not have an explicit mechanism for relational reasoning between objects and backgrounds, potentially impairing its ability to identify duplicates and compare attributes from different image regions.

Consistent with our claim that complex questions require more than detection, DETECT is the worst performer on Test-Complex. DETECT performs better on Test-Simple, but there is still a large gap between it and RCN.

To study the importance of the object-background model, we ran RCN without the $RN(O, B)$ component. As seen in Table 4, this hurts performance for both simple and complex questions showing the value of the background model.

(a) How many giraffes are there?
GT: 2, DETECT: 2, Zhang:2, RCN: 2

(b) How many people are standing?
GT: 2, DETECT: 4, Zhang: 3, RCN: 2

(c) How many people in the front row?
GT: 8, DETECT: 22, Zhang: 6, RCN: 8

(d) How many chairs have a girl sitting on them?
GT: 1, DETECT: 7, Zhang: 2, RCN: 1

(e) How many players are wearing red uniforms?
GT: 3, DETECT: 11, Zhang: 4, RCN: 3

(f) How many strings does the instrument to the left have?
GT: 4, DETECT: 3, Zhang: 1, RCN: 0

Figure 4: Example model outputs on TallyQA. While other models fail at positional reasoning questions (*e.g.* Fig. 5c), RCN can infer an object's relative position to other objects. Since RCN is based on region proposals, it struggles when proposals do not align with question relevant objects (Fig. 5i).

Positional Reasoning Questions. Since RCN uses object-based relational reasoning, we expect it to outperform other methods for positional reasoning questions. To study this, we filtered out positional reasoning questions from TallyQA's Test-Complex set using common qualifiers such as *left, right, top, up, bottom, near, on, in*, and then we measured accuracy for Zhang *et al.* and RCN. We found that RCN outperformed Zhang *et al.* 's model by 6.38% absolute for these questions, which further demonstrates RCN's efficacy.

Performance Without Location Features

To assess the impact of using the spatial location information of each proposal, we conducted an experiment in which we removed the location features s_{ij} given to RCN. For HowMany-QA, removing location caused a 5.4% decrease in accuracy (absolute). For TallyQA, it caused a decrease of 2.8% accuracy (absolute) for Test-Simple and 2.4% accuracy (absolute) for Test-Complex.

Comparison with the Original RN

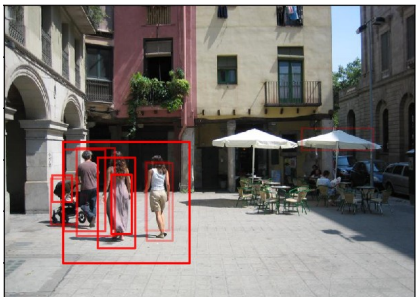
The original RN model uses raw CNN feature maps, rather than region proposals. Running the original RN model on HowMany-QA, it achieved 3.46 RMSE and about 20% less accuracy (absolute) than RCN. RCN likely achieves better performance due to its improved architecture and due to using region proposals.

Visualizing RCN

To visualize RCN's inference process, we modified Grad-CAM (Selvaraju et al. 2017). Grad-CAM is a technique that, for a given prediction, generates a coarse heat map based on the gradient flow in the final convolutional layers. To adapt Grad-CAM to RCN, it is necessary to derive scores for each proposal. To do this, we first find the pairwise object-background score $score(o_i, b_j)$ using the gradient obtained at layer g_{θ_2}. We then assign a score to each proposal using $score(o_i) = \max_{i,j} score(o_i, b_j)$. Scores for all proposals are then scaled from 0 to 1 and visualized on the original image. Examples are shown in Fig. 5.

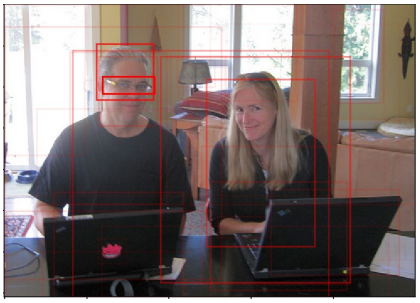
Discussion

RCN achieved state-of-the-art results across all of the datasets, even outperforming Zhang *et al.*, which is the best published result on VQA2's counting questions, and IRLC, which was the previous best result on HowMany-QA. The same regions and visual features were used across RCN, Zhang *et al.* , and IRLC, so the difference in performance is not due to using superior visual features, which is a frequent confound in many works. Our experiments showcased that there is a large performance gap between the ability for models to answer simple and complex questions. This gap was especially large for RCN and the Zhang *et al.* method. A likely reason is that more data is required for complex questions to handle the full range of attributes and relations.



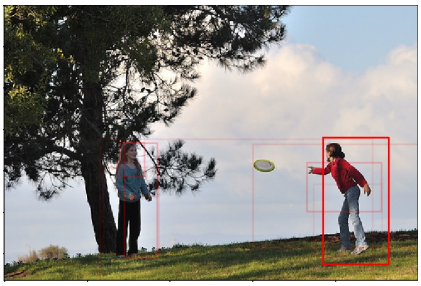
(a) How many people are wearing long dresses?

(b) How many people are sitting on a horse?

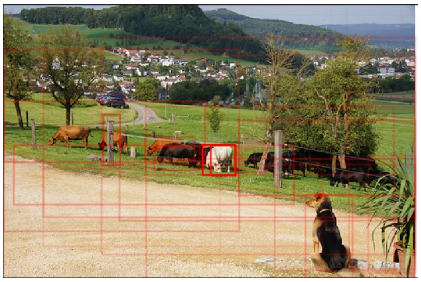


(c) How many people are wearing glasses?

(d) How many people have a hat?

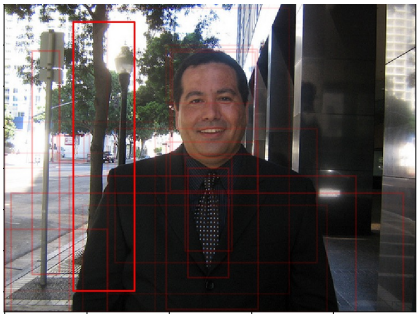


(e) How many players are wearing red?

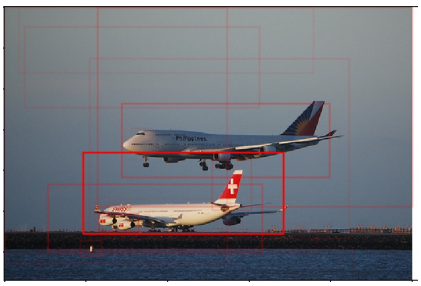


(f) How many white cows are there?

(g) How many dogs are sleeping in the image?



(h) How many street lights are to be seen behind this man?



(i) How many of the planes are on the ground?

Figure 5: Modified Grad-CAM visualizations show where RNC is looking to make predictions. The importance of each object proposals is proportional to the color intensity of the bounding boxes.

We found that region based methods, such as RCN, IRLC, and the Zhang *et al*. model have better results compared to direct methods, e.g., MUTAN (Ben-younes et al. 2017). However, all of these region based models, including ours, are not based on actual *nameable* objects but *object proposals*, which range from 10–100 in number for each image and can consist of many non-object regions and overlapping boxes. Intelligently pruning/refining of these proposals may improve performance of these systems. We tried simple non-maximal suppression to prune out the overlapping boxes for RCN, but it did not improve performance. We believe this to be due to the relational capacities of RCN which can learn to ignore duplicate or similar boxes based on the features and positions of the boxes more intelligently than off-the-shelf non-maximal suppression.

Conclusions

In this paper, we distinguished between simple and complex open-ended counting questions in VQA, where simple questions could be correctly answered using object detection alone. To do this, we created TallyQA, the world's largest dataset for open-ended counting using VQA, which will be made publicly available. We also described the RCN framework and showed that it can effectively answer both simple and complex counting questions compared to baseline models and state-of-the-art approaches for open-ended counting. RCN combines region proposals with relationship networks, enabling them to be efficiently used with high-resolution imagery. We found that RCN worked especially well compared to others on complex questions. Our work better defines the issues with open-ended counting, and sets the stage for future work on this problem.

Acknowledgments

We thank Robik Shrestha for useful discussions. The lab thanks NVIDIA for the donation of a GPU, and we thank Alexander Trott for assistance with HowMany-QA.

References

[Anderson et al. 2018] Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; and Zhang, L. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*.

[Andreas et al. 2016] Andreas, J.; Rohrbach, M.; Darrell, T.; and Klein, D. 2016. Learning to compose neural networks for question answering. In *NAACL*.

[Antol et al. 2015] Antol, S.; Agrawal, A.; Lu, J.; Mitchell, M.; Batra, D.; Zitnick, C. L.; and Parikh, D. 2015. VQA: Visual question answering. In *ICCV*.

[Ben-younes et al. 2017] Ben-younes, H.; Cadene, R.; Cord, M.; and Thome, N. 2017. Mutan: Multimodal tucker fusion for visual question answering. In *ICCV*.

[Chattopadhyay et al. 2017] Chattopadhyay, P.; Vedantam, R.; RS, R.; Batra, D.; and Parikh, D. 2017. Counting everyday objects in everyday scenes. In *CVPR*.

[Dalal and Triggs 2005] Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *CVPR*.

[Fukui et al. 2016] Fukui, A.; Park, D. H.; Yang, D.; Rohrbach, A.; Darrell, T.; and Rohrbach, M. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding.

[Girshick 2015] Girshick, R. 2015. Fast R-CNN. In *ICCV*.

[Goyal et al. 2017] Goyal, Y.; Khot, T.; Summers-Stay, D.; Batra, D.; and Parikh, D. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *CVPR*.

[He et al. 2016] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

[Johnson et al. 2017] Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Zitnick, C. L.; and Girshick, R. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.

[Kafle and Kanan 2016] Kafle, K., and Kanan, C. 2016. Answer-type prediction for visual question answering. In *CVPR*.

[Kafle and Kanan 2017a] Kafle, K., and Kanan, C. 2017a. Visual question answering: Datasets, algorithms, and future challenges. *CVIU*.

[Kafle and Kanan 2017b] Kafle, K., and Kanan, C. 2017b. An analysis of visual question answering algorithms. In *ICCV*.

[Kafle et al. 2018] Kafle, K.; Cohen, S.; Price, B.; and Kanan, C. 2018. DVQA: Understanding data visualizations via question answering. In *CVPR*.

[Kafle, Yousefhussien, and Kanan 2017] Kafle, K.; Yousefhussien, M.; and Kanan, C. 2017. Data augmentation for visual question answering. In *INLG*.

[Kim, Jun, and Zhang 2018] Kim, J.-H.; Jun, J.; and Zhang, B.-T. 2018. Bilinear attention networks. *arXiv preprint arXiv:1805.07932*.

[Lu et al. 2016] Lu, J.; Yang, J.; Batra, D.; and Parikh, D. 2016. Hierarchical question-image co-attention for visual question answering. In *NIPS*.

[Malinowski and Fritz 2014] Malinowski, M., and Fritz, M. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *NIPS*.

[Pennington, Socher, and Manning 2014] Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*.

[Redmon and Farhadi 2017] Redmon, J., and Farhadi, A. 2017. YOLO9000: better, faster, stronger. In *CVPR*.

[Ren and Zemel 2017] Ren, M., and Zemel, R. S. 2017. End-to-end instance segmentation with recurrent attention. In *CVPR*.

[Ren et al. 2015] Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*.

[Ren, Kiros, and Zemel 2015] Ren, M.; Kiros, R.; and Zemel, R. 2015. Exploring models and data for image question answering. In *NIPS*.

[Ryan et al. 2009] Ryan, D.; Denman, S.; Fookes, C.; and Sridharan, S. 2009. Crowd counting using multiple local features. In *Digital Image Computing: Techniques and Applications, 2009. DICTA'09.*, 81–88. IEEE.

[Santoro et al. 2017] Santoro, A.; Raposo, D.; Barrett, D. G.; Malinowski, M.; Pascanu, R.; Battaglia, P.; and Lillicrap, T. 2017. A simple neural network module for relational reasoning. In *NIPS*.

[Selvaraju et al. 2017] Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D.; et al. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*.

[Teney et al. 2017] Teney, D.; Anderson, P.; He, X.; and Hengel, A. v. d. 2017. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *arXiv preprint arXiv:1708.02711*.

[Trott, Xiong, and Socher 2018] Trott, A.; Xiong, C.; and Socher, R. 2018. Interpretable counting for visual question answering. In *ICLR*.

[Wang and Wang 2011] Wang, M., and Wang, X. 2011. Automatic adaptation of a generic pedestrian detector to a specific traffic scene. In *CVPR*.

[Zhang et al. 2015] Zhang, C.; Li, H.; Wang, X.; and Yang, X. 2015. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*.

[Zhang et al. 2016] Zhang, P.; Goyal, Y.; Summers-Stay, D.; Batra, D.; and Parikh, D. 2016. Yin and yang: Balancing and answering binary visual questions. In *CVPR*.

[Zhang, Hare, and Prügel-Bennett 2018] Zhang, Y.; Hare, J.; and Prügel-Bennett, A. 2018. Learning to count objects in natural images for visual question answering. In *ICLR*.

An Automated Question-Answering Framework Based on Evolution Algorithm

Sinan Tan[1], Hui Xue[2], Qiyu Ren[3], Haiping Liu[1] and **Jing Bai[4]**
[1]Tsinghua University, China
[2]Microsoft Research, China
[3]University of Posts and Telecommunications, China
[4]Microsoft, Silicon Valley, U.S

Abstract

Building a deep learning model for a Question-Answering (QA) task requires a lot of human effort, it may need several months to carefully tune various model architectures and find a best one. Its even harder to find different excellent models for multiple datasets. Recent works show that the best model structure is related to the dataset used, and one single model cannot adapt to all tasks. In this paper, we propose an automated Question-Answering framework, which could automatically adjust network architecture for multiple datasets. Our framework is based on an innovative evolution algorithm, which is stable and suitable for multiple dataset scenario. The evolution algorithm for search combine prior knowledge into initial population and use a performance estimator to avoid inefficient mutation by predicting the performance of candidate model architecture. The prior knowledge used in initial population could improve the final result of the evolution algorithm. The performance estimator could quickly filter out models with bad performance in population as the number of trials increases, to speed up the convergence. Our framework achieves 78.9 EM and 86.1 F1 on SQuAD 1.1, 69.9 EM and 72.5 F1 on SQuAD 2.0. On NewsQA dataset, the found model achieves 47.0 EM and 62.9 F1.

1. Introduction

Question-Answering (QA) is a key problem in the field of artificial intelligence. It requires one to find the correct answer for a given question from passages. In our work, we focus on the task of Reading Comprehension, where the question is grounded on related documents or passages. Due to the diversity of language expression and the complexity of grammar, understanding the semantic of long passages and queries has always been a difficult task. Recently, a large number of deep learning models including BiDAF (Seo et al. 2016), R-Net (Wang et al. 2017), FusionNet (Huang et al. 2017) and QANet(Yu et al. 2018) are proposed by researchers, and they do achieved good performance on certain datasets (e.g. SQuAD dataset(Rajpurkar et al. 2016)).

However, one single model cannot adapt to all the tasks. For example, (Joshi et al. 2017) shows that a model that performs well for SQuAD may fail to get outstanding result on TriviaQA. Therefore, for different reading comprehension datasets, different models may need to be designed, which will take a lot of human effort and time.

In this work, we propose a new evolution algorithm based method to build an automated Question-Answering framework. Leveraging neural architecture search (NAS) and transfer learning, we are able to design suitable model architectures for different reading comprehension tasks, requiring significantly less time and human effort.

Most existing algorithms (Zoph and Le 2016; Zoph et al. 2017; Pham et al. 2018; Liu, Simonyan, and Yang 2018; Tan et al. 2018) for neural architecture search, are optimized for a predefined model with strong knowledge of domain experts. They focus on designing a directed acyclic graph (DAG) as a cell structure for the model, and transfer their knowledge between different tasks by sharing and repeating same basic cell over and over again. However, from the previous work(Seo et al. 2016; Wang et al. 2017; Huang et al. 2017; Yu et al. 2018), most state-of-art models in reading comprehension consists of a variety of complex connections, and do not contain any largely repeated model structure. Therefore it is almost impossible to design a predefined global structure for reading comprehension models.

In addition, the existing methods like ENAS(Pham et al. 2018), based on reinforcement learning and using s deep learning model as a controller, are not suitable for finding model architectures for multiple datasets. Due to the instability of reinforcement learning(Mnih et al. 2015), different datasets may require different hyper-parameters, which means human intervention is actually necessary. So it's hard to apply these methods for various datasets.

We build a flexible framework to search for Reading Comprehension models. We do not predefine any high-level structure for the models. In other words, model architectures can grow dynamically in our framework. Besides, our method, which uses evolution algorithm, is insensitive to hyper-parameters. It is very suitable for the scenario of multiple datasets.

Previous research work(Real et al. 2017; Jaderberg et al. 2017; Real et al. 2018) have shown that evolutionary algorithm are feasible to find a neural network architecture for image classification. Our method is also based on evolution algorithm. Compared to image recognition, the Reading Comprehension task is very different. The models in Reading Comprehension usually need to use Recurrent Neural

Networks(RNN) layers to encode the semantics and attention layers to compute the relationship between query and document, so the models suitable for Reading Comprehension are more complex and have longer training time. This means the search time of evolution might be very long.

In order to solve this problem, we propose two main ideas. First, the models used to initialize population in our evolution algorithm come from the state-of-art models and diverse stochastic models mutated from those models. In traditional evolution algorithms(Real et al. 2017), a large number of experiments in the early stages are needed to explore and get a better direction of mutation, which is very inefficient. Searching from the state-of-art models could introduce human knowledge into the search progress and improve the final result in limited time, which could be concluded from our experiments. Second, we use a performance estimator to estimate the performance of candidate models and try to model the relationship between performance and model architectures. The concept of using neural network as performance predictor has been proposed in Peephole (Deng, Yan, and Lin 2017). When performing the next mutation, the performance estimator will be used to evaluate the performance of several different possible mutated models and our framework could pick best candidate to train. This way, we can avoid inefficient random mutation, and speed up the search progress.

Here is the summarization of our contributions in the proposed framework:

- We build an automated Question-Answering framework based on innovative evolution algorithm. Its a flexible framework and can automatically adapt to different Reading Comprehension datasets and produce competitive results.
- We propose an improvement based on naive evolution algorithm(Real et al. 2017), which is initializing population from the state-of-art models. This can improve the performance of the found model.
- We use a CNN-based performance predictor to estimate the performance of candidate mutations, which is a more efficient alternative to naive random mutation. This can accelerate the search progress and speed up its convergence.

2. Related Work

Our work is highly related to neural architecture search (NAS) and Reading Comprehension. Here we will summarize previous work in the two fields.

2.1 Neural Architecture Search.

NAS(Zoph and Le 2016) uses a RNN controller to discover neural network architectures by searching for an optimal graph to maximize the expected accuracy of the generated architectures on a validation set. NAS takes a lot of time for getting a single model. Besides, ENAS(Pham et al. 2018) speeds up the search progress by sharing parameters among the models being searched. (Cai et al. 2018) use Net2Net transfer (Chen, Goodfellow, and Shlens 2015) to speed up the search when updating the network structure.

But these methods are not suitable for the multiple datasets scenario in reading comprehension for two reasons. First, the methods based on reinforcement learning or deep learning models, are very sensitive with their hyper-parameters and thus aren't) suitable for the multiple dataset scenario. Second, unlike image recognition, known Question-Answering models are very complex. Most competitive QA models do not contains a large number of repeated basic blocks similar to ResNet (He et al. 2016a). Therefore, it's impossible to find a cell structure suitable for every dataset. Our framework is inspired by(Real et al. 2017), which proposes a paralleled evolution algorithm, by randomly picks two models from the population and randomly mutates the better one. But there are two drawbacks in original evolution. One is random mutation, which is ineffective and uncontrollable. The other is that the evolution algorithm consumes a large amount of resources to filter the weak individual compared to other methods. Our method uses known state-of-art models to initialize the population, and uses performance estimator to avoid inefficient mutation and speed up the convergence.

2.2 Reading Comprehension.

There are many systems for reading comprehension that employ embedding at the character level and word level. (Chen et al. 2017) applied some linguistic features to input embedding. (Peters et al. 2018) introduce a new word embedding called ELMo, which is learned by training a bidirectional language model. (McCann et al. 2017) use a deep LSTM encoder trained for machine translation as contextualized word vectors called CoVe. Adding ELMo and CoVe could improve the performance of the common NLP tasks, by using information from large datasets. In our paper, we take advantage of these embeddings as the input layer in our automated QA framework. (Wang and Jiang 2016) propose match-LSTM to predict answer boundaries with pointer network. The pointer network has also become a common method for the model of reading comprehension. We use pointer-network as the output layer for predicting the answer span. (Seo et al. 2016) use use bi-directional attention flow mechanism to obtain a query-aware context representation. (Wang et al. 2017) proposed a self-matching attention mechanism to refine the representation by matching the passage against itself, which effectively encodes information from the whole passage. (Huang et al. 2017) proposed a fully-aware multi-level attention mechanism to capture the complete information in one text (such as a question) and exploit it in its counterpart (such as context or passage) layer by layer. (Yu et al. 2018) proposed use convolution and self attention to encode text, where convolution models local interactions and self-attention models global interactions.

These papers focus on modeling the interactions between the context and the query. But for different datasets, the interaction may complex and have no fixed pattern. (Joshi et al. 2017) showed that BiDAF performs well in SQuAD 1.1 but failed in TriviaQA, which means different tasks need different model. In our paper, we leverage these research works and use those proposed layers to design the search space for QA model. As for the connections between those layers, we

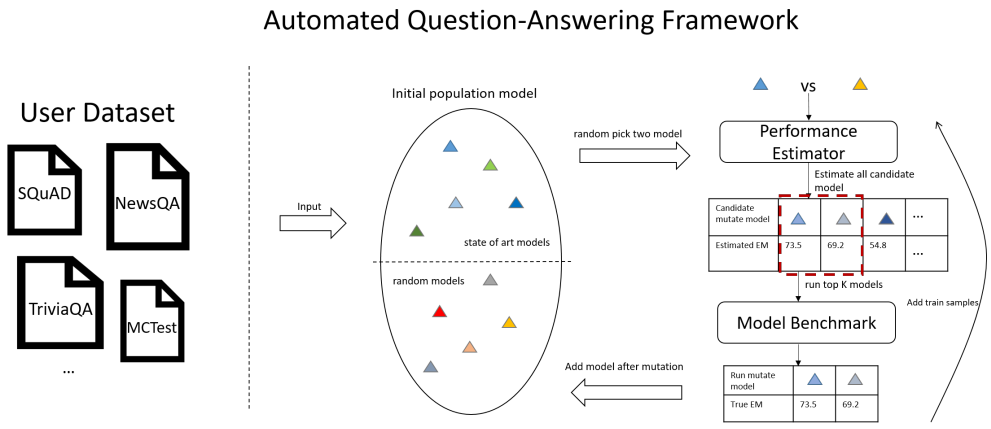


Figure 1: Overall architecture of our framework

use the proposed evolution algorithm based neural architecture search to find the best solution.

3. An Automated Question-Answering Framework

Figure 1 shows the pipeline of our Question-Answering framework. Here is an overview of our proposed framework: Before being fed into the model, all documents and questions in a dataset will be tokenized and prepossessed to extract linguistic features. Then the state-of-art of model architecture in QA and diverse random model architectures mutated from them will be added into initial populations. Following the traditional evolution algorithm(Real et al. 2017), the framework will randomly pick two individual(models) from populations. We then choose the model that achieves better performance before mutation. We propose an optional variant of the traditional mutation as an alternative to random mutation. It uses a CNN-based performance predictor to estimate the performance of models after possible mutations. We will keep the history of the model structures having been trained in the process of evolution, and train the performance prediction model with the structure of those QA models and their performance.

In the following sections, we will explain details for different components in our framework.

3.1 Initial Population

For better use of prior knowledge, we consider adding state-of-art models into initialized population, including BiDAF (Seo et al. 2016), RNET (Wang et al. 2017) and FusionNet (Huang et al. 2017) and so on. It is also possible to add even more known models to the framework, but we are going to stick with those three models in our experiments.

Models designed by humans beings could be considered as a local optima for neural architecture structure. Using them speeds search but may also limit our search space. Therefore, other models, obtained by performing a large number of mutations on those known models are also added to the initial population, which could help to guarantee the diversity of population.

3.2 Search Space

Table 1 shows the statistics about the structures of other state-of-art models. Based on that result, we use the following mutation actions in our proposed framework.

- IDENTITY (Effectively means keep training).
- INSERT-RNN-LAYER (Inserts a LSTM. Comparing the performance of GRU and LSTM in our experiment, we decided to use LSTM here.)
- REMOVE-RNN-LAYER
- INSERT-ATTENTION-LAYER(Inserts a attention layer. Comparing with other attention, we consider use symmetric attention proposed by FusionNet (Huang et al. 2017))
- REMOVE-ATTENTION-LAYER
- ADD-SKIP (Identity between random layers).
- REMOVE-SKIP (Removes random skip).
- CONCAT-TWO-INPUT(Combines output of two layers into one using concat operation)

The layers contained in search space were chosen for their similarity to the actions that a human designer may take when improving an architecture. The probabilities for these mutations are equal before they are estimated by the performance estimator.

3.3 Performance Estimator

In the search progress, we will generate several valid candidate models with the mutation space mentioned above. And then our framework will use the performance estimator to estimate the EM score of each candidate model.

To estimate the score of each candidate model, here we use CNN instead of LSTM to build the prediction model. The model configuration will be represented by an adjacency matrix of the computation graph. We believe that CNN is able to capture shared structures in the neural network, as those structures will exhibit locality in adjacency matrix.

Model Name	RNN	Attention	Skip Connection	Conv	Concat
BiDAF(Seo et al. 2016)	Yes	Yes	No	No	No
DCN(Xiong, Zhong, and Socher 2016)	Yes	Yes	No	No	Yes
ReasoNet(Shen et al. 2017)	Yes	Yes	No	No	No
R-net(Wang et al. 2017)	Yes	Yes	No	No	Yes
FusionNet(Huang et al. 2017)	No	Yes	Yes	Yes	Yes
QANet(Yu et al. 2018)	Yes	Yes	No	No	No

Table 1: Building blocks of known QA models.

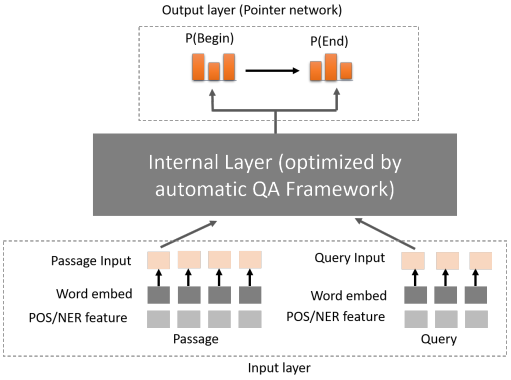


Figure 2: Skeleton of our models

An adjacency matrix can be embedded to an 2D feature map, which have been proven can be effectively processed by convolution neural network. Each adjacency matrix's shape is $N \times N \times K$, where N is the maximum number of nodes the model may contain, and a K-dimensional embedding is used to represent the relation of nodes.

The embedding of relation for the layers are related to the search space, for example, if node X is the output of another node Y, after getting through an LSTM layer, then the cell in the X-th row and Y-th column of the feature map will be the embedding vector of the relation "is input of LSTM", and the cell in the Y-th row and X-th column of the feature map will be marked to the embedding vector of the relation "is output of LSTM". The way of encoding attention layer, concatenate layer, and the skip connections is also similar to this. "No connection", "Self", "Padding" are considered as special relation types, and are also added to the feature map matrix.

We use a PreResNet50 (He et al. 2016b) as the model to predict the performance of the reading comprehension model in our framework. The output layer is changed to be normalized by sigmoid function: $\frac{1}{1+e^{-x}}$, representing the exact match score.

We use the following L2 loss for the training procedure.

$$L(\theta) = (r - r^*)^2 \tag{1}$$

where θ is parameters of the performance estimator, r is the predicted performance of the model, $r*$ is the actual performance of the model.

During the search progress, an increasing number of models will be trained, which means more and more training data will be available for the performance estimator. Therefore, the probability (p) of the model with better predicted performance chosen will change by time:

$$p = 0.5 + 0.5 * \frac{min(epoch, epoch_{max})}{epoch_{max}} \tag{2}$$

Where $epoch_{max}$ is a constant after which number of epochs model with better predicted performance will always be chosen, $epoch$ is the number of model trained. The intuition is that, as the number of models being trained increasing, the performance estimator become more and more reliable. So as $epoch$ is increasing, $epoch$ is also expected to increase, the probability of adopting the result proposed by the performance estimator is higher.

The use of this performance estimator is optional. We analyze how it affects the training procedure in the last section of the paper.

3.4 Model Configuration

The model configuration in our framework contains three parts: input layer, internal architecture, and output layer. The input layer and output layer are fixed in our framework. internal architecture is only part we try to tune. Model Benchmark is a toolkit that will parse these configurations and compile them to a model that could run in a deep learning platform, after which we get the performance of this model.

- Input Layer

  For the word embedding, we concatenate the pre-trained 300-dimensional GloVe vectors (Pennington, Socher, and Manning 2014), 1024-dimensional ELMo vectors (Peters et al. 2018) and 600-dimensional CoVe vectors (McCann et al. 2017). Following (Chen et al. 2017), we use three additional types of linguistic features for each word : 1) a 44-dimensional POS tagging embedding, 2) an 13-dimensional NER tagging embedding, and 3) a 3-dimensional binary exact match feature. The character level embedding is the same as(Wang et al. 2017), taking the final hidden states of a bi-directional RNN applied to embedding of characters in the token. We concatenate all these embeddings as input layer in our framework for queries and documents.

- Internal Architectures

  Internal Architectures is the key part in our framework. The internal architectures is generated from the configuration and in the process of evolution.

The layers in internal architecture are always from the search space.

- LSTM
  LSTM(Hochreiter and Schmidhuber 1997) is a common type of RNN widely used in language models. In our experiments, models with LSTM outperform those with GRU (Cho et al. 2014). So we use LSTM instead of GRU to extract language features.
- Attention
  We use the symmetric attention function proposed by (Huang et al. 2017), where the scoring function is

$$s = f(Ux)^T D f(Uy)^T \tag{3}$$

  We replaced the non-linearity function f with Swish activation function (Ramachandran, Zoph, and Le 2017), because we found that Swish activation speeds the training process of a single model.
  The above attention scoring function is used for both self-attention and document-to-question or question-to-document attention.
- Concatenate Layer
  Following (Huang et al. 2017), we use the concatenate layer to fuse different level of concepts is helpful for the model. Here we simply introduce this operation to our framework, which concatenate the feature dimension of document or question vector.
- Skip-Connection Layer
  Skip connection is very common in very deep convolution neural networks, and is used by (Yu et al. 2018). Whether or not it works for RNN based architecture is still uncertain, but considering the potential benefit, we still add it to our framework.

- Output layer
  We follow (Wang et al. 2017) and use pointer networks (Wang and Jiang 2016) to predict the start and end positions of the answer. We use an avg-pooling over the query representation to generate u^Q for the pointer network.

$$u^Q = avg\_pool(\{u_t^Q\}_{t=1}^n) \tag{4}$$

  Then we attend for the span start using the summarized question understanding vector u^Q.

$$\begin{aligned} s_i^t &= V^S tanh(W^S u_i^P + W^Q u^Q) \\ P_i^S &= \frac{exp(s_i^t)}{\sum_{i=1}^n exp(s_j^t)} \end{aligned} \tag{5}$$

  To use the information of the span start, we combine the context understanding vector for the span start with u^Q through a GRU cell (Cho et al. 2014).

$$\begin{aligned} v^Q &= GRU(u^Q, \sum_{i=1}^n P_i^S u_i^P) \\ e_i^t &= V^E tanh(W^E u_i^P + W^Q u^Q) \\ P_i^E &= \frac{exp(e_i^t)}{\sum_{i=1}^n exp(e_j^t)} \end{aligned} \tag{6}$$

  During training, we minimize the sum of the negative log probabilities of the ground truth start and end positions in the predicted distributions. During prediction, we keep the answer span pos^s, pos^e with maximum $P_{pos^s}^S P_{pos^e}^E$ and $pos^s \leq pos^e - 15$. For datasets with questions impossible to answer, a special "no answer" vector is append to the document sequence. When predicted pos^s points to this vector, the problem is considered unanswerable.

The Pointer Network is always used as the output layer in our framework. Figure 2 shows the skeleton of our model.

3.5 Algorithm Detail

After the above discussion about the details of our framework, we show the modified version of evolution algorithm used in our framework.

Algorithm 1 Evolution-based model search with performance estimator

$N \leftarrow$ *population size*
$PI \leftarrow$ *existing models*
$M \leftarrow$ *performance estimator update frequency*
$P \leftarrow PI$
for $i \in 0 : (N - len(PI))$ **do**
    $PI\_sample \leftarrow PI.random\_choice()$
    $P \leftarrow P + PI\_sample.multiple\_mutate()$
for $w \in workers$ **do**
    $i, j = sample(P)$
    if $performance(i) > performance(j)$ **then**
        Delete *j* from P
        $k \leftarrow mutate(i)$
    else
        Delete *i* from P
        $k \leftarrow mutate(j)$
    $p = 0.5 + 0.5 * \frac{min(epoch, epoch_{max})}{epoch_{max}}$
    if $random(0.0, 1.0) > p$ **then**
        $k \leftarrow k.random\_choice()$
    else
        $k \leftarrow k.choice\_with\_value\_function()$
    if $total\_models\_trained()\%M == 0$ **then**
        Update performance estimator model
    Train k and put it into population

4. Experiments

We test our framework on three different datasets: NewsQA (Trischler et al. 2016), SQuAD 1.1 (Rajpurkar et al. 2016), SQuAD 2.0 (Rajpurkar, Jia, and Liang 2018).

Unlike other existing datasets like MSMARCO (Nguyen et al. 2016),these datasets focus exclusively on questions with answers that can be found in the document. NewsQA is a Reading Comprehension dataset containing over 100,000 human-written question-answer pairs, whose documents are from CNN/Daily Mail. Comparing with SQuAD 1.1, the number of question-answer pairs of NewsQA is slightly smaller but the average length of query and documents is much longer. SQuAD 2.0 is much more harder dataset with

Single Model	Exact Match	F1
FastQA(Weissenborn, Wiese, and Seiffe 2017)	68.4	77.1
BiDAF(Seo et al. 2016)	68.0	77.3
SEDT(Liu et al. 2017b)	68.2	77.5
RaSoR(Liu et al. 2017a)	70.8	78.7
FastQAExt(Weissenborn, Wiese, and Seiffe 2017)	70.8	78.9
ReasoNet(Shen et al. 2017)	70.6	79.4
DrQA(Chen et al. 2017)	70.7	79.4
R-net(Wang et al. 2017)	75.7	83.5
FusionNet(Huang et al. 2017)	75.3	83.6
QANet(Yu et al. 2018)	76.2	84.6
EvolutionRC	**78.9**	**86.1**
R.M-Reader(Hu et al. 2017)	78.9	86.3
SLQA+(Wang, Yan, and Wu 2018)	80.0	87.0

Table 2: The performance of EvolutionRC and competing models on SQuAD 1.1 development set.

over 50,000 unanswerable questions written adversarially by crowd-workers. Those unanswerable questions are produced by looking up similar answerable ones. (Rajpurkar, Jia, and Liang 2018) shows a strong neural model for SQuAD 1.1 dataset suffers from a huge drop when transfer to SQuAD 2.0.

4.1 Experiment Setup

For the evolutionary algorithm, we set the initial population size to 32. The maximum number of layers is limited to 50. The number of workers is 8, which means we run the architecture search in 8 GPUs in parallel. In these experiments, we do not use the performance estimator to speed up the training procedure.

For the SQuAD v1.1 and SQuAD v2.0 datasets the GPU we used is Tesla P100-PCIE. For NewsQA dataset, whose average context length is much longer, we use Tesla P40-PCIE, which has 24GB of RAM in total.

For data preprocessing, we use the tokenizer from the Stanford CoreNLP (Manning et al. 2014) to preprocess each passage and question and to extract part-of-speech tagging (POS) and named entity recognition (NER) features. We also apply dropout (Srivastava et al. 2014) before all RNN layers, after embedding layers, and before linear transform in attention cell. The dropout rate for RNN and embedding layers is 0.5. The dropout rate for attention cell is 0.3.

The model is optimized by Adamax (Kingma and Ba 2014), with initial learning rate $lr = 2 \times 10^{-3}$, $\beta = (0.9, 0.999)$, and $\epsilon = 1 \times 10^{-8}$.

We train each model for 50 epochs. Also we use early stopping to avoid overfitting during the trial experiments. The max patience is 5 epoch. In addition, the patience will increase when the performance is still increasing with numbers very closed to patience of epochs are trained.

For each dataset, we stop the experiment after 500 models are trained. This costs about one week for 8 GPUS, or about 50 GPU days. However, we believe that it is possible to use lower dropout rate and downsampled datasets to significantly lower the computation cost.

The batch size is 64 for the SQuAD v1.1 and SQuAD v2.0 datasets, and is 32 for NewsQA dataset. When searching for the architecture, we only use CoVe feature. However, we use both CoVe and ELMo feature when retraining the found model.

We use the official evaluation script for all datasets. We use (Microsoft Research Asia 2018) as our experiment platform.

4.2 Main Results

Using our proposed framework for reading comprehension, we performed architecture search on SQuAD 1.1, SQuAD 2.0, and NewsQA Dataset. We report average exact match and F1 scores for our model after retrained with ELMo embedding. The searched model achieves 78.9 EM and 86.1 F1 on SQuAD 1.1, 69.9 EM and 72.5 F1 on SQuAD 2.0. On NewsQA dataset, the found model achieves 47.0 EM and 62.9 F1. We also compare the model searched by our method with other models designed by humans in Table 2, Table 3, and Table 4.

For SQuAD 1.1 and SQuAD 2.0, our best model is able to outperform many published state-of-art models. But the searched model still cannot reach the performance of top models in the SQuAD competition leaderboard.

We believe that those model might be using more domain knowledge, external data sources, more techniques for data augmentation, or different ways of predicting the range for the answers, which is beyond the scope for our proposed framework. Besides, we didn't tune hyper-parameters for our best model.

5. Discussion

There are two key points in our proposed framework: initializing population from existing models, and guiding the evolution process with a value-function based performance estimator. To understand how they affect the process of evolution algorithm, We conduct ablation studies on these components of the proposed framework, including the proposed initialization method, and the performance estimator.

Single Model	Exact Match	F1
BNA(Rajpurkar, Jia, and Liang 2018)	59.8	62.6
DocQA(Rajpurkar, Jia, and Liang 2018)	61.9	64.8
DocQA + ELMo(Rajpurkar, Jia, and Liang 2018)	65.1	67.6
EvolutionRC	**69.9**	**72.5**

Table 3: The performance of EvolutionRC and competing models on SQuAD 2.0 development set.

Single Model	Exact Match	F1
Match-LSTM(Wang and Jiang 2016)	34.4	49.6
BARB(Trischler et al. 2016)	36.1	49.6
BiDAF (Seo et al. 2016)	37.1	52.3
FastQA(Weissenborn, Wiese, and Seiffe 2017)	43.7	56.4
FastQAExt(Weissenborn, Wiese, and Seiffe 2017)	43.7	56.1
EvolutionRC	**47.0**	**62.9**
AMANDA(Kundu and Ng 2018)	48.4	63.3

Table 4: The performance of EvolutionRC and competing models on NewsQA development set.

5.1 Initialization Method

To understand how the initialization method affect the final result of evolution algorithm, we run the framework on SQuAD dataset again, with completely random initialization. All other experiment setup is kept the same. Here is the table showing the difference. With the proposed initialization method, the EM score increases from 76.0 to 78.9.

Single Model	Exact Match	F1
EvolutionRC	78.9	86.1
Naive Evolution(Real et al. 2017)	76.0	84.1

Table 5: The effect of initializing population with known models.

5.2 Performance Estimator

To investigate how the CNN-based performance estimator affects the training procedure, we perform experiments on SQuAD dataset, with and without the performance estimator. When running this experiment, we initialize the population completely by random. The evolution process stops after 400 models are trained. We choose $epoch_{max} = 100$ for the performance estimator in this experiment.

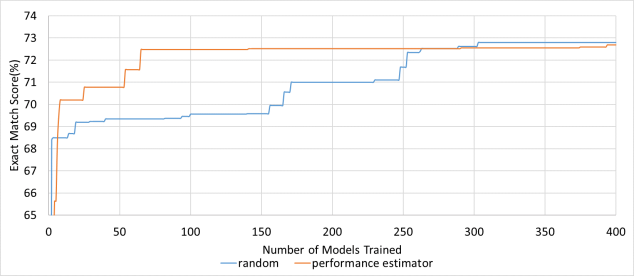


Figure 3: Convergence of evolution process

From the above figure, we can see the performance estimator speeds up the convergence of the evolution algorithm by avoiding ineffective mutations. However, the performance estimator does not improve the final result, so we didn't use it in the experiments mentioned in Section 4. The experiment is repeated only once, so it might be affected by random factors.

6. Conclusion

In this paper, we propose a new evolution algorithm based framework for different Question-Answering problems. We propose a new variant of evolution algorithm, which uses a CNN-based performance estimator, and existing human-designed models as starting points for the mutation process. Our experiment shows that (i) This framework achieves near state-of-art performance on multiple datasets with limited human intervention and acceptable computational resources. (ii) The proposed way of initializing population is better than random initialization, in terms of final result. (iii) The use of the proposed performance estimator speeds up the convergence of the search progress.

References

Cai, H.; Chen, T.; Zhang, W.; Yu, Y.; and Wang, J. 2018. Efficient architecture search by network transformation. AAAI.

Chen, D.; Fisch, A.; Weston, J.; and Bordes, A. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Chen, T.; Goodfellow, I.; and Shlens, J. 2015. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Deng, B.; Yan, J.; and Lin, D. 2017. Peephole: Predicting network performance before training. *arXiv preprint arXiv:1712.03351*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645. Springer.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hu, M.; Peng, Y.; Huang, Z.; Qiu, X.; Wei, F.; and Zhou, M. 2017. Reinforced mnemonic reader for machine reading comprehension. *arXiv preprint arXiv:1705.02798*.

Huang, H.-Y.; Zhu, C.; Shen, Y.; and Chen, W. 2017. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*.

Jaderberg, M.; Dalibard, V.; Osindero, S.; Czarnecki, W. M.; Donahue, J.; Razavi, A.; Vinyals, O.; Green, T.; Dunning, I.; Simonyan, K.; Fernando, C.; and Kavukcuoglu, K. 2017. Population Based Training of Neural Networks. *arXiv preprint arXiv:1711.09846*.

Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kundu, S., and Ng, H. T. 2018. A question-focused multi-factor attention network for question answering. *arXiv preprint arXiv:1801.08290*.

Liu, R.; Hu, J.; Wei, W.; Yang, Z.; and Nyberg, E. 2017a. Structural embedding of syntactic trees for machine comprehension. *arXiv preprint arXiv:1703.00572*.

Liu, X.; Shen, Y.; Duh, K.; and Gao, J. 2017b. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. DARTS: Differentiable Architecture Search. *arXiv preprint arXiv:1806.09055*.

Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; and McClosky, D. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 55–60.

McCann, B.; Bradbury, J.; Xiong, C.; and Socher, R. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, 6294–6305.

Microsoft Research Asia. 2018. Nni. `https://github.com/Microsoft/nni`.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Nguyen, T.; Rosenberg, M.; Song, X.; Gao, J.; Tiwary, S.; Majumder, R.; and Deng, L. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Rajpurkar, P.; Jia, R.; and Liang, P. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Ramachandran, P.; Zoph, B.; and Le, Q. V. 2017. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*.

Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y. L.; Tan, J.; Le, Q.; and Kurakin, A. 2017. Large-Scale Evolution of Image Classifiers. *arXiv preprint arXiv:1703.01041*.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2018. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Shen, Y.; Huang, P.-S.; Gao, J.; and Chen, W. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1047–1055. ACM.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; and Le, Q. V. 2018. Mnasnet: Platform-aware neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*.

Trischler, A.; Wang, T.; Yuan, X.; Harris, J.; Sordoni, A.; Bachman, P.; and Suleman, K. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Wang, S., and Jiang, J. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Wang, W.; Yang, N.; Wei, F.; Chang, B.; and Zhou, M. 2017. Gated self-matching networks for reading comprehension and question answering. 189–198.

Wang, W.; Yan, M.; and Wu, C. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1705–1714.

Weissenborn, D.; Wiese, G.; and Seiffe, L. 2017. Making neural qa as simple as possible but not simpler. *arXiv preprint arXiv:1703.04816*.

Xiong, C.; Zhong, V.; and Socher, R. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Yu, A. W.; Dohan, D.; Luong, M.-T.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Zoph, B., and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2017. Learning transferable architectures for scalable image recognition. *CoRR* abs/1707.07012.

Natural Language Question Answering over BI data using a Pipeline of Multiple Sequence tagging Networks

Mrinal Rawat, Pratik Saini, Amit Sangroya, C. Anantaram, Gautam Shroff
TCS Innovation Labs, Tata Consultancy Services Limited, ASF Insignia, Gwal Pahari, Gurgaon, India
(rawat.mrinal, pratik.saini, amit.sangroya, c.anantaram, gautam.shroff)@tcs.com

Abstract

Enterprise data is usually stored in the form of relational databases. However, accessing this data currently requires users to understand a query language such as SQL. Business analysts more often are not familiar with the syntax of SQL and they often struggle with data science tools to get data insights. State of art tools do not have an agent driven natural language based conversational interface that can interact with user and provide necessary data insights. Moreover, State of art tools have limited capabilities to understand user's context and intentions.In this paper, we propose a deep learning based framework that generates a database query from a natural language sentence. Interesting feature of our approach is to focus upon domain agnosticity. Our framework is based on a set of machine learning models that create an intermediate sketch from a natural language query. Using the intermediate sketch, we generate a final database query over a large knowledge graph. Our framework supports multiple queries such as aggregation, self joins, factoid and transnational. We evaluate the proposed approach over widely popular WikiSQL dataset and two enterprise datasets. Our results show that even with limited training data, model is able to capture the user's semantics well. The system is designed for usability such that even naive business analysts can use the system comfortably.

Introduction

Various enterprise applications such as finance, retail, pharmacy etc. store a vast amount of data in the form of relational databases. However, accessing relational databases requires an understanding of query languages such as SQL, which, while powerful, is not easy for business analysts to master. Natural language interfaces (NLI), a research area at the intersection of natural language processing and human-computer interactions, seeks to provide means for humans to interact with computers through the use of natural language. We explore one aspect of NLI applied to relational databases; creating a data science assistance framework that can be used by business analysts to get data insights using natural language interactions.

Building a data science assistant that captures context and semantic understanding is an important and challenging problem. The problem is multi-dimensional as it involves complexities at multiple levels e.g. relational database level, natural language interface level and semantic understanding level. At relational database level, the system must be able to handle the complexities related to data representation. At natural language interface level, the system should support NL related issues such as handling ambiguity, variations and semantics. Most importantly, the system must be able to understand user's context and should respond back in a meaningful way.

There are a number of recent works which are trying to build Conversational AI systems that support human-like cognitive capabilities such as context-awareness, personalization, and ability to handle complex NL inputs. Apple Siri, Microsoft Cortana, Amazon Alexa are some of the widely used personal assistants that are already in market. However, there are still a variety of open research issues while designing conversation AI based digital assistant tools (Jadeja and Varia 2017). To address the limitations of existing techniques towards designing conversational AI based personal digital data assistants, we make following key contributions in this paper:

- We present `Curie`, a data science assistant that supports NL conversational interface for effective data assistance across a range of tasks i.e. NL question answering and dialogue, data visualization and data transformation. The Curie platform is designed to support context, intent identification, multiple ways of interaction through dialog, question answering and a variety of NL query processing capabilities.
- Secondly, we also introduce a novel approach for parsing natural language sentences and generating an intermediate representation (query sketch) using a combination of machine learning models. This intermediate representation is further used to execute low level database queries.
- We demonstrate our results on two enterprise datasets and one public dataset i.e. WikiSQL. Our results show state of art performance on these datasets while supporting transfer learning that ensures that our approach can be directly used across a variety of domains with minimal effort.

Related Work

There are a lot of conversational AI based personal assistant tools already in market for the end users. Some of these

tools are designed for open-domain conversations e.g. Cortana, Siri, Alexa and Google Now (Sadun and Sande 2013; Ehrenbrink, Osman, and Möller 2017). On the other hand there are also tools where focus is on supporting interactive data assistance e.g. Microsoft Power BI (Parks 2014). However, there are still limitations of the existing solutions in terms of natural language based conversational capabilities while supporting data assistance. There are still various open issues in terms of semantic understanding and context-awareness user's queries. Most important task for building a NL based data assistant is to generate structural query language (SQL) queries from natural language. Answering a natural language question about a database table requires modeling complex interactions between the columns of the table and the question. Jedeja et al. present four different perspectives namely user experience, information retrieval, linguistic and artificial intelligence for the evaluation of conversational AI systems (Jadeja and Varia 2017).

The study of translating natural language into SQL queries has a long history. Popescu et al. introduce *Precise NLI*, a semantic parsing based theoretical framework for building reliable natural language interfaces (Popescu, Etzioni, and Kautz 2003). This approach rely on high quality grammar and is not suitable for tasks that require generalization to new schema. Recent works consider deep learning as the main technique. There are many recent works that tackle the problem of building natural language interfaces to relational databases using deep learning (Xu, Liu, and Song 2017; Yaghmazadeh et al. 2017; Zhong, Xiong, and Socher 2017; Yin et al. 2015; Pasupat and Liang 2015; Li and Jagadish 2014; Yu et al. 2018; Dong and Lapata 2018). Zhong et al. (Zhong, Xiong, and Socher 2017) propose *Seq2SQL* approach that uses reinforcement learning to break down NL semantic parsing task to several sub-modules or sub-SQL incorporating execution rewards. Yavuz et al. introduce *DialSQL*, a dialogue based structured query generation framework that leverages human intelligence to boost the performance of existing algorithms via user interaction (Yavuz et al. 2018). The flexibility of our approach enables us to easily apply sketches to a new domain. Our framework also does not require large corpus of NL sentences as training input.

Curie: Data Science Assistant

Curie is designed as a set of intelligent suite of AI tools designed to help business analysts in getting insights about data in a user friendly way. Instead of meandering through the database for a small detail, Curie provides an interface where business analysts just need to type their query in Natural Language (English) and system will present the results. Curie supports context understanding by precisely capturing user's intent and responding appropriately.

More often, business analysts have a range of queries that they perform to get data insights (See Figure 1). Sometimes they just need a one line answer for their questions such as *"What is phone number of Julia"*. Sometimes it might be some aggregation query e.g. *"How many employees are hired in 2018"* and in some cases they might be interested in visualizing the information in the form of charts e.g. *"Show me the acceptance rate of EMNLP over last ten years"*. Therefore, Curie is designed to handle all these variations of NL queries. It supports a conversational interface that business analysts can use and get their information in fraction of seconds. It makes interactions with data quick, efficient, and easy. Depending on the nature of query, Curie can also respond with appropriate visualized interpretation of the data in the form of Pie Charts, Graphs etc.

Table 1: Variety of Natural Language Queries

Transactional	How many employees are in Mumbai?
FAQ	How can I apply sick leave?
Factoid	What is the population of Mars?
Aggregations	How many employees in each project?
Visualization	Tell me the monthly profit of company in last decade?
Sorting	List all employees based on their age

We propose an architecture where a database query is formed from a natural language sentence with the help of an intermediate form i.e. query sketch. Our system comprises of two parts: a) Mechanism to generate an intermediate form (**Query Sketch**) given a NL sentence and b) Approach of transforming a sketch to database query.

Deep Learning based Framework for Query Sketch Generation

In order to generate the query sketch, we have a pipeline of multiple sequence tagging deep neural networks. Our architecture consists of a bidirectional LSTM network alongwith a CRF (conditional random field) output layer. In our architecture framework, the sequence of word embedding is given as input to a bidirectional LSTM. Instead of using the softmax output from this layer, we use a CRF Layer yielding the final predictions for every word (See Figure 2). We use ELMO embedding that are computed on top of two-layer bidirectional language models with character convolutions as a linear function of the internal network states (Peters et al. 2018). The character-level embedding have been found useful for specific tasks and to handle the out-of-vocabulary problem. The character-level representation is then concatenated with a word-level representation and feed into the Bidirection LSTM as input. The intent is to identify the parts in the sentence which are relevant. We annotate a NL query as follows. *{What is the employee id of John}* is annotated as {0 0 0 A A 0 B} (See Figures 3 and 4 for reference schema and example sequence respectively).

This example shows that there are two concepts *employee id* and *John* marked as A and B respectively. The same token is used for the concepts which consist of more than one word e.g. *employee id* consists of two words, so we mark them with token 'A'. We follow this representation in our models. This approach makes our system database agnostic and even with comparatively less amount of training data we are able to extract the information out of the sentence. Since models are dependent on each other, we will explain each of them with the help of an example: *How many employees work in project Curie?*

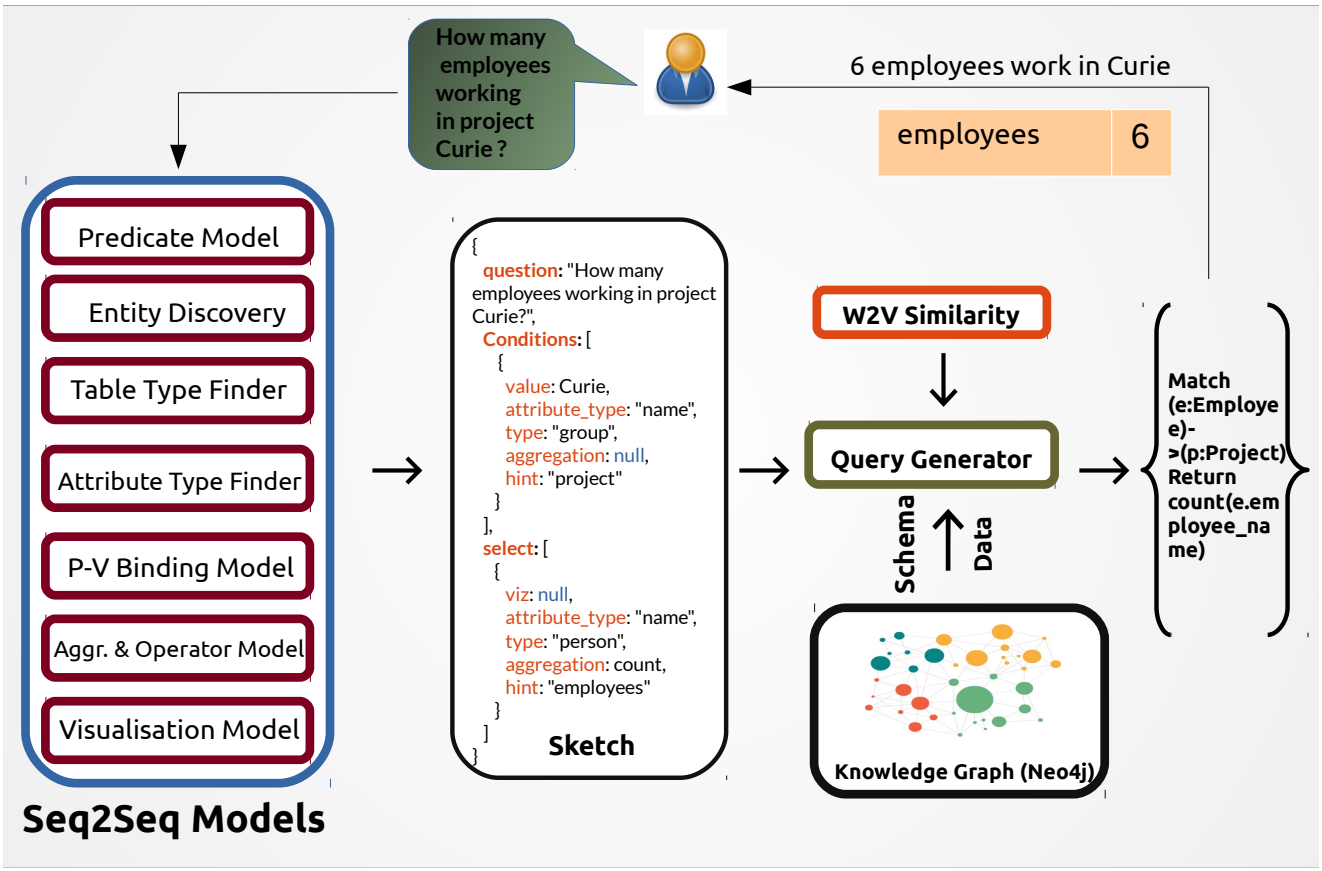


Figure 1: Models for NL to Query generation

Predicate Finder Model This model finds the target concepts (**predicates**) P from the NL sentence. In case of database query language, predicate refers to the `SELECT` part of the query. Once predicates are identified, it becomes easier to extract entities from the remaining sentence. The input to the model is a vector form representation of NL sentence. For example, a natural language sentence, *{How many employees work in project Curie}* is annotated as {0 0 A 0 0 B 0}. In this example *employees* and *project* are predicates.

Entity Discovery Model This model aims to find the **values/entities** in the sentence. The output from predicate model is taken and reformed as: *{How many <predicate> work in <predicate> Curie}*. The predicates are replaced with < ***predicate*** > token.

We assume that structured data for the domain is present in `Apache-Solr`. Thereafter, we discover the entities in the reformed sentence using `Lucene`. This is explained as follows. Firstly, the part-of-speech tags are extracted from the input. Thereafter, we ignore part-of-speech tags which are stop words. Now, using a sliding window we prepare N-grams from the remaining string. For each N-gram, we do a search using `Apache Lucene`. N-grams which correspond to an entity in the data and having the highest score are picked. The detailed approach for this model is explained using **Algorithm 1**. For the above mentioned example, remaining string would be: *work Curie*. Hence, *Curie* is picked as an entity.

Type Level Finder This model identifies the **type of concepts** (predicates and values) at the node or table level. For example, *employee id* belongs to *Employee* node and *Employee* is a *PERSON*, so the type of *employee id* is *PERSON*.

If a concept is present in more than one table, **type** information helps in the process of disambiguation. For example, consider following sentences:

- What is the employee id of Washington?
- List all the stores in Washington.

Here, in first example *Washington* refers to the name of a person, whereas in second example it is the name of a location. In such cases, this model is useful to disambiguate that in first example the node level type is `PERSON` and in second example it is `LOCATION`. This helps in making the overall framework database agnostic. In this model, all the entities in input are marked with tag <*value*>. For example, natural language sentence, *{How many employees work in project <value>}* is annotated as {0 0 person 0 0 project project}.

Attribute Level Type Finder This model identifies the **attribute** type of concepts (predicates and values). For example, let's take two sentences:

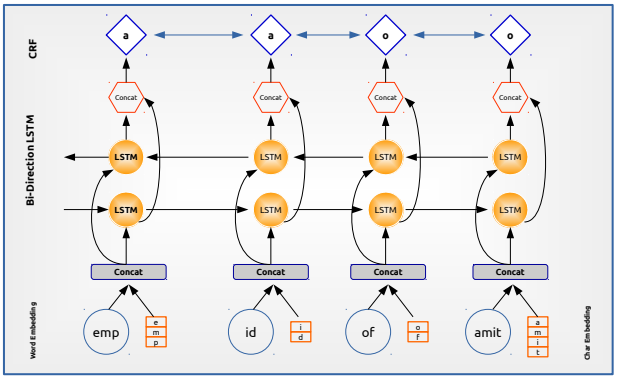


Figure 2: Main architecture of the Seq Tagging network. Word embedding and character embedding are concatenated and given to a bidirectional LSTM. First layer represents the word and the left context, second layer represents the word and its right context. These two vectors of bi-directional LSTM are concatenated and are projected into CRF layer which finally yields the prediction of the every word.

- What is the employee id of May?
- List all employees hired in May?

In these examples, *May* is present in the same table *Employee*, but refers to different attributes. The attribute type information model here can easily distinguish based on the nature of query that in first example, *May* is the name of a person and in second example it is *date*. The attribute type information in combination with node type information helps in increasing the accuracy. In this model, all the entities in input are marked with tag *<value>*. For example, *{How many employees work in project <value>}* is annotated as: {0 0 name 0 0 name name}.

Predicate Value Finder In some queries **predicate value bindings** are already present. For example, let's take two examples:

- How many employees work in Curie?
- How many employees work in project Curie?

In first example, there is no information about the concept *Curie*, but the second example describes that *Curie* is some project. This model binds the predicate, in this example *project* to the value or entity *Curie*. We replace the predicates with tag *<predicate>* and entities with tag *<value>*. *{How many <predicate> work in <predicate> <value>}* is annotated as: {0 0 0 0 0 A A}. The tokens at index 5 and 6 are binded as [*Project*] ?? [*Curie*]. "??" slot will be filled with an operator that we illustrate in next model.

Aggregations & Operators Finder In this model, **aggregations and operators** are predicted for predicates and entities respectively. As explained earlier, our framework currently supports following set of aggregation functions: count, groupby, min, max, sum, and sort. Similarly, following set of operators are also supported: $=, <, >, <>, >=, <=$. *{How many <predicate> work in <predicate> <value>}* is annotated as: {0 0 count 0 0 0 equal}. We can infer that predicate *project* and entity *Curie* are related as [*Project*] = [*Curie*]. Finally, using the output from all these LSTM based deep learning models, following intermediate form (query sketch **S**) is generated.

procedure `ExtractEntity`(Q, T)**:**
  Input: natural language query Q, stop/unwanted pos tags T
  Output: A list of entities E
  $PosTags \leftarrow$ TAGGER(Q);
  $S \leftarrow \emptyset$; `// Remaining string`
  foreach *postag* $p_i \in PosTags$ **do**
    if $p_i \notin T$ **then**
      $S.add$(p)
    end
  end
  $NGrams \leftarrow$ findNgrams(S);
  $E \leftarrow \emptyset$; `// List of entities`
  while $i < length(NGrams)$ **do**
    $c \leftarrow$ Q(NGrams(i));
    $maxScore \leftarrow 0$;
    $e \leftarrow \emptyset$;
    foreach *integer* $k \in (i, length(NGrams))$ **do**
      $c.add$(Q(NGrams(k)));
      $s \leftarrow$ luceneSearch(c);
      if $s > maxScore$ **then**
        $maxScore \leftarrow s$;
        $e.add$(Q(NGrams(k)));
      end
    end
    if $e \neq null$ **then**
      $E.add$(e);
    end
  end
  return E;

Algorithm 1: Finding the entities from the NL sentences

```
Select:{
        pred_hint:= employees
        type:= PERSON
        attr_type:= name
        aggregation:= count
}
Conditions:{
        pred_hint:= project
        value:= Curie
        type:= PROJECT
        attr_type:= name
}
```

Generating Database Query from Sketch

The process of generating the query is independent of underlying database i.e. the same approach can be used for generating queries across databases. We demonstrate this con-

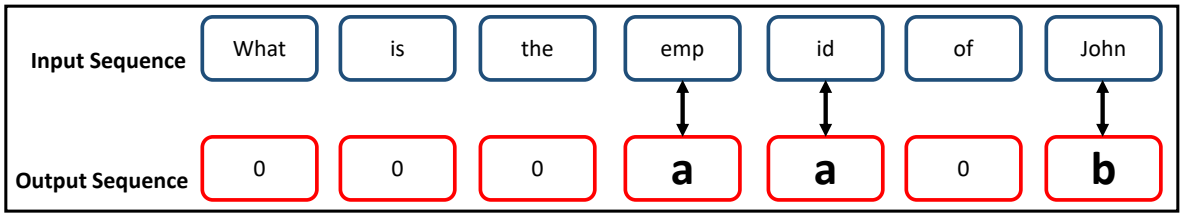


Figure 3: Example of Sequence tagging for an Input Sequence. Output Sequence shows tag for predicate part.

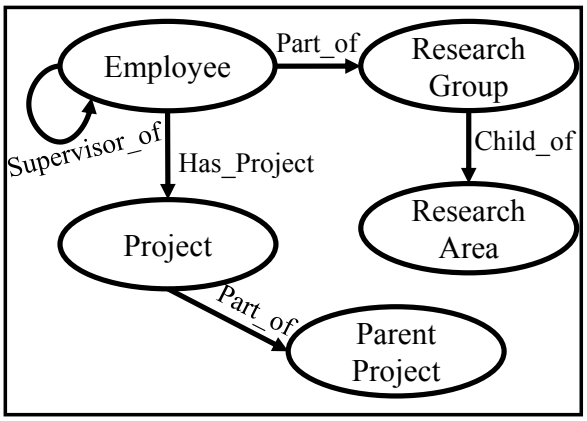


Figure 4: Schema example of enterprise data

cept using two popular relational database query languages: SQL (structured query language) and CQL (cipher query language). The general idea to generate a query is as follows. Sketch **S** is a set {**P,C**}, where **P** is a set {$p_1, p_2, p_3, \ldots p_n$} and each p_i is {n_i, a_i, h_i, g_i}, where
n_i = Node/table level type of predicate p_i
a_i = Attribute/predicate level type of predicate p_i
h_i = NL Hint for the predicate p_i
g_i = Aggregation information for the predicate p_i
Similarly, **C** is a set {$c_1, c_2, c_3, \ldots c_n$} where each c_i is {n_i, a_i, h_i, v_i, o_i}
n_i = Node/table level type for the predicate involved in condition c_i
a_i = Attribute/attribute level type for the predicate involved in condition c_i
h_i = NL hint for the predicate involved in condition c_i
g_i = Value involved in condition c_i
o_i = Operator involved in condition c_i

Provided with a sketch **S**, we generate a database query **Q**. As explained earlier, sketch contains information about all the required predicates **L**, i.e. the list of attributes/columns that need to be extracted from Table **T**. This component will give a new relation that only contains the columns in **L**. This operation is projection (Π) in relation algebra. The sketch has also information about all conditions φ which gives us a new relation that contains only those rows satisfying φ in **T**. This is the selection (σ) operation of relation algebra. To generate the query following set of operations are executed over the sketch:

- Analysing the predicate part (P) of the sketch:
  For all predicates $p_i \in$ P do:
  - Use the NL hint h_i to check semantic similarity with all the predicates of table type n_i and attribute type a_i to get the closest matching predicate p
  - Find the table t of the predicate p of the table type n_i
  - **L** = {p | p ∈ P}
- Analysing the condition part (C) of the sketch:
  For all the condition $c_i \in$ C do:
  - Use the value v_i to perform EDL and select the best candidate using NL hint for predicate h_i , attribute type a_i and table type n_i to find the predicate p for the value v_i
  - Find the table t of the predicate p of the table type n_i
  - φ = {p o v | p,o,v ∈ P}

The next step is to find all the unique tables (U) involved in a given sketch **S**. If the len(U) > 1, we find the shortest path passing through all the unique tables. We assume that there is a path if the foreign key relationship exists between tables. This step will give us join ($\bowtie$) operation of relation algebra. Operator and aggregation information are already present in the sketch S. Now, combining these, we compute the final relation (**R**). Using **R**, we finally compute the DB specific query (See **Algorithm 2** for details). For example, a natural language sentence *{"How many employees in each project"}* leads to following database query:

$$\Pi_{count(employee\_id),g(project\_name)} (Employee_{pid\_fk} \bowtie_{pid} Project)$$

a. Database Query SQL:

```
SELECT COUNT(emp_id), proj_name
FROM Employee INNER JOIN Project
ON (pid_fk = pid) GROUP BY proj_name
```

b. Database Query CQL:

```
MATCH
  (e:Employee)-[:WORKS]->(p:Project)
RETURN
  COUNT (e.employee_id), p.project_name
```

Results and Discussion

We conducted our experiments on an Intel Xeon(R) computer with E5-2697 v2 CPU and 64GB memory, running Ubuntu 14.04. We evaluated Curie on three real-world datasets,

```
procedure QueryGenerator(S, D):
    Input: Sketch S,
    Database Information D
    N ← ∅;                          // List of nodes
    foreach select s_i ∈ S.select do
        S.predicate ← w2vSimilarity(s_i, D);
        // Semantic similarity with
        all the properties in actual
        graph
        N.add(node)
    end
    foreach condition c_i ∈ S.conditions do
        node ← EntityDiscovery(c_i.value);
        // Find the nodes in which
        value is present
        if node ≠ null then
            N.add(node)
        end
    end
    ShortestPath ← FindShortestPath(N);
    Q ← ∅;                                  // Query
    if ShortestPath ≠ null then
        Q.match ← = ShortestPath;
        foreach condition c_i ∈ S.conditions do
            Q.where ← c_i;
        end
        foreach select s_i ∈ S.select do
            Q.return ← s_i;
        end
        Query ← ConstructQuery(Q);
        R ← FetchFromDB(Query);      // Fetch
        result from db
        return R;
    end
    else
        return "No Matching Data";
    end
```

Algorithm 2: Generating the DB Query from Sketch S

Table 2: Datasets and their statistics

Dataset	# Tables	# Columns	Train. set	Test set
Ent. Dataset-1	5	42	1700	300
Ent. Dataset-2	3	70	1700	300
WikiSQL	24241	1542564	56355	15878

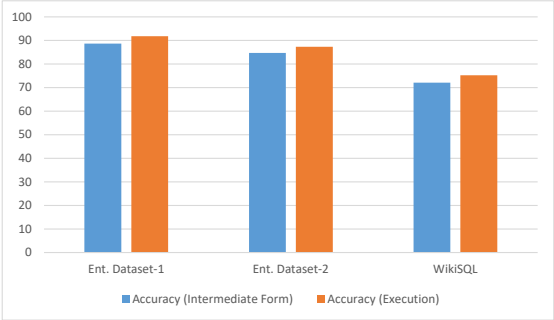


Figure 5: Overall accuracy across Datasets

out of which two are our internal enterprise datasets (See Table 2). First dataset is related to employees in a large software services company and their allocations. Our second internal dataset is about a large pharmaceutical company's stores and employee's information. [1] Our third dataset for experimental evaluation is widely used WikiSQL dataset. Ent. Dataset-1 & 2, composed of following type of queries: factoid, self joins, Group By, sorting, and aggregations. WikiSQL dataset is composed of factoid, aggregation. Queries in WikiSQL were only over single columns.

Note that, in order to generate the intermediate form(sketch), all models are not necessary. The predicate, value, aggregation & operator models are sufficient to generate the sketch. The type, attribute and PV binding models are just used to improve the accuracy. As seen in Table 3 and Figure 5, Curie has shown a very good performance for Ent. Dataset-1 & 2. Recent works have reported approx. 80% accuracy on WikiSQL dataset (Guo et al. 2018; Sun et al. 2018). Our results also demonstrate similar results with simple network architecture. Our entity discovery model was based on the `Lucene` search. It could be further improved by combining `Lucene` search with deep learning based entity extraction. Interestingly, for evaluating Ent. Dataset-2, we just used the same model that was trained on Ent. Dataset-1. We can see that almost similar results were achieved on the latter dataset demonstrating the transfer learning capabilities of our ML models.

[1] These datasets are available for download and public use at `https://github.com/nlpteam19/curie`.

Table 3: Accuracy across models for all datasets

	Ent. Dataset-1	Ent. Dataset-2	WikiSQL
Predicate Acc.	94.6	93.9	91.4
Value Acc.	94.56	92	88.2
Type Acc.	84	82.86	NA
Attribute Acc.	80	78.2	NA
PV Binding Acc.	92.57	92.34	NA
Aggr. & Operator Acc.	93.1	93.4	92
Overall Sketch Acc.	86.64	84.74	72.1
Overall Exec. Acc.	91.78	87.3	75.24

Conclusion and Future Work

We proposed Curie as a novel framework for performaing natural language question answering over BI data. Our approach is based on deep learning using multiple sequence tagging networks and knowledge graph that uses minimal training data and supports data assistance across multiple domains. Our framework captures the user context and provides a robust conversational interface for getting insights in eneterprize data. In future, we plan to explore and evaluate multi-tasking capabilities i.e. having an intermediate representation and supporting a range of other tasks.

References

Dong, L., and Lapata, M. 2018. Coarse-to-fine decoding for neural semantic parsing. *CoRR* abs/1805.04793.

Ehrenbrink, P.; Osman, S.; and Möller, S. 2017. Google now is for the extraverted, cortana for the introverted: Investigating the influence of personality on ipa preference. In *Proceedings of the 29th Australian Conference on Computer-Human Interaction*, OZCHI '17, 257–265. New York, NY, USA: ACM.

Guo, D.; Sun, Y.; Tang, D.; Duan, N.; Yin, J.; Chi, H.; Cao, J.; Chen, P.; and Zhou, M. 2018. Question generation from sql queries improves neural semantic parsing. *CoRR* abs/1808.06304.

Jadeja, M., and Varia, N. 2017. Perspectives for evaluating conversational ai. *CoRR* abs/1709.04734.

Li, F., and Jagadish, H. V. 2014. Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.* 8(1):73–84.

Parks, M. 2014. *Microsoft Business Intelligence. POWER BI*. USA: CreateSpace Independent Publishing Platform.

Pasupat, P., and Liang, P. 2015. Compositional semantic parsing on semi-structured tables.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. *CoRR* abs/1802.05365.

Popescu, A.-M.; Etzioni, O.; and Kautz, H. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, 149–157. New York, NY, USA: ACM.

Sadun, E., and Sande, S. 2013. *Talking to Siri: Learning the Language of Apple's Intelligent Assistant*. Que Publishing Company, 2nd edition.

Sun, Y.; Tang, D.; Duan, N.; Ji, J.; Cao, G.; Feng, X.; Qin, B.; Liu, T.; and Zhou, M. 2018. Semantic parsing with syntax- and table-aware sql generation. In *ACL*.

Xu, X.; Liu, C.; and Song, D. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning.

Yaghmazadeh, N.; Wang, Y.; Dillig, I.; and Dillig, T. 2017. Sqlizer: Query synthesis from natural language. *Proc. ACM Program. Lang.* 1(OOPSLA):63:1–63:26.

Yavuz, S.; Gur, I.; Su, Y.; and Yan, X. 2018. Dialsql: Dialogue based structured query generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, 1339–1349.

Yin, P.; Lu, Z.; Li, H.; and Kao, B. 2015. Neural enquirer: Learning to query tables with natural language.

Yu, T.; Li, Z.; Zhang, Z.; Zhang, R.; and Radev, D. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 588–594. Association for Computational Linguistics.

Zhong, V.; Xiong, C.; and Socher, R. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning.

Step Semantics: Representations for State Changes in Natural Language

Ken Forbus[1], Maria Chang[2], Danilo Ribeiro[1],

Tom Hinrichs[1], Maxwell Crouse[1], Michael Witbrock[2]

[1]Qualitative Reasoning Group, Northwestern University, 2233 Tech Drive, Evanston, IL, 60208
[2]IBM TJ Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, NY 10598
forbus@northwestern.edu

Abstract

The dynamics of the world is often bound up in processes. These include continuous processes, such as flows and motion, and discrete processes, such as count and break. Things that occur in the world can often be described at multiple levels of detail, using combinations of continuous and discrete processes, and it is important to be able to shift among levels of detail as needed for communication and understanding. This paper describes *step semantics,* a framework that draws upon prior work in qualitative reasoning and discrete action representations to provide a set of representation conventions for processes described in natural language, independent of a particular task or dataset. We explore its potential in two ways: Analyses of recipes with complex temporal structure and learning from AI2's ProPara dataset.

Introduction

Human level complex question answering requires deep understanding of processes and procedures. These processes can include continuous quantities, like speed, or discrete quantities, like integer counts. Moreover, processes and their sub events are often described at different levels of detail. For example, "cook dinner" can be viewed as a discrete event, but it can involve many instances of continuous processes (e.g. mixing, splitting, heating, cooling) when viewed at a finer level of detail. Similarly, the life cycle of a frog might be described in terms of three discrete states: eggs, tadpoles, and adults, even though the growth of legs in a tadpole and the shrinkage of its tail happen smoothly over many days. Question-answering systems need to be able to represent both discrete and continuous processes and reason about them in ways that are compatible with each other.

Although considerable advances have been made in reasoning for question-answering, understanding processes is still a major challenge. Few datasets include questions that require inference about processes, and most are in the domain of science tests, e.g. ARC (Clark et al. 2018) & ProPara (Dalvi et al. 2018). These datasets are steps in the right direction, but there are more subtle phenomena that they do not test, as explained below.

This paper presents a representation for processes described in text that combines qualitative process theory (Forbus, 1984) with models of discrete actions and change from OpenCyc and FrameNet to go beyond what either could do alone. We show the utility of this synthesis by examining both recipes, which can incorporate complex temporal structure and combinations of continuous processes and discrete actions, and learning from AI2's ProPara dataset. We argue that this synthesis provides a prerequisite for human-level reasoning for answering questions about processes.

Background & Related Work

To provide a set of representation conventions for processes described in natural language, we draw upon prior work in qualitative reasoning and discrete action representations, which we summarize here.

Qualitative Process Theory

Qualitative process theory is a representational system for describing continuous processes. Processes provide a notion of mechanism, in that, aside from the actions taken by agents, ultimately all changes are explainable in terms of the effects of processes. This strong inductive bias simplifies learning and conceptual change (e.g. Friedman et al. 2017). Liquid flow, for example, happens between a source and destination. Its direct effects – direct influences – are specified as part of the process. For example, in liquid flow,

```
(I+ (AmountOf ?dest) (FlowRate ?lf))
(I- (AmountOf ?src) (FlowRate ?lf))
```

That is, the amount of liquid in the source is decreased by the flow rate of the liquid flow, and the amount of liquid in the destination is increased by the same rate. Processes are *active* when their conditions are satisfied, e.g. for liquid flow, when the pressure in the source is greater than the pressure in the destination. Continuous processes are typically expressed in language via verbs, e.g. flow, move. Participants are typically described in language via role information about the verb, including prepositional phrases. For example, "Water flowed out of the bathtub onto the floor." describes a liquid flow whose source is the bathtub and whose destination is the floor. Notice that the path is implicit: This is a common property of language, we tend to leave implicit things that are not important or are inferable by the listener.

Causal laws associated with objects support inferring the indirect effects of processes via *qualitative proportionalities*, which are partial information about functional dependencies. For example, in a contained liquid (Hayes, 1984),

```
(qprop+ (Level ?l) (AmountOf ?l)
(qprop+ (Pressure ?l) (Level ?l))
```

That is, a change in amount will cause a change in level, which in turn will cause a change in pressure.

Quantities in QP theory are described in terms of ordinal relationships with other quantities, where the relevant set of comparisons is automatically derived from the structure of the domain theory. For example, a liquid which has a fluid path to other liquids will lead to their relative pressures being tracked, because that is part of what determines if liquid flow is active. If phase changes such as freezing and boiling are under consideration, the temperature of the liquid will also be compared with its melting and boiling points. These *limit points* are often mentioned in texts, e.g. "When all the water is drained from the pasta…" While sometimes specific numerical values are known (e.g. "Cook the roast until its internal temperature is 165 degrees."), often they are not (e.g. "Wait until the mixture has cooled.").

Qualitative representations carve time up into discrete units, based on when qualitative properties change. Following Hayes (1984), we represent changes over time in terms of *histories*, which are pieces of space-time over which the qualitative properties of some set of objects is the same. For instance, the cooking episode in the history of the creation of a roast starts when the roast is placed in the oven and ends when it is removed. Its spatial aspect is the union of the spatial aspects of the participants in it, e.g. the oven and the roast. We note that, in many qualitative reasoning projects, a more global notion of qualitative state is often used, where all of the entities under consideration are lumped together. We prefer using histories here because they allow for finer-grained decomposition of behavior that seems more suitable to the level of partial information found in language.

Events and Discrete Actions

To represent events, we draw upon a combination of concepts from FrameNet (Ruppenhofer et al. 2016) and the OpenCyc ontology. Specifically, we use neo-Davisonian representations, where events are reified and role relations are used to describe their particular aspects, such as participants, location, and duration. For example, consider the word "convert". In Cyc conventions, the word itself is denoted by an entity (i.e. `Convert-TheWord`). FrameNet has four senses of convert when used as a verb, which draw on three semantic frames (i.e. `FN_Undergo_transformation`, `FN_Cause_change`, and `FN_Exchange_currency`). Each sense is also linked in the KB to an event from the Cyc ontology (i.e. `Converting-Something`, `Convincing-CommunicationAct`, `CurrencyExchange`, `IntrinsicStateChange`). The FrameNet information provides two valuable sources of information for supporting natural language understanding. The first is a mapping from lexemes (i.e. word senses) to frames. For example, eight lexemes evoke the `FN_Creating` frame. The second are a set of *valence patterns* that help constrain parsing by stating what patterns of auxiliary phrases are common. The OpenCyc information provides semantic constraints, including type information, allowable role relations, and inference rules concerning that type of event.

We assume events take time, although for some perspectives, that time is so short that it can safely be treated as an instant (Allen & Hayes, 1990). Events whose internals are irrelevant to understand a particular text can be considered as discrete actions. To provide the inferential semantics for discrete actions, we assume STRIPS operators (Fikes & Nilsson, 1971) for simplicity.

We note that in the Qualitative Reasoning community, there have been several prior efforts that integrate discrete and continuous models of actions and processes, albeit for very different purposes. Hogge (1987) described how QP descriptions of processes could be compiled into operators for use with a temporal planner. Forbus (1989) explored how STRIPS operators could be added to envisionments based on QP theory, to simulate systems that incorporated actions alongside physical processes. Drabble (1993) showed how QP theory could be combined with an HTN planner to both generate and execute plans involving both actions and processes. None of these prior efforts addressed integrating continuous and discrete representations in understanding natural language, which is our focus here.

Answering Questions about Processes

Reading comprehension is largely evaluated through question answering tasks. State of the art performance on these tasks is generally achieved using artificial neural networks that take a query and context (e.g. a paragraph) as inputs and

predict a span of text within the context that contains the answer (e.g. Chen et al. 2017, Seo et al. 2017). However, by definition this poses a challenge when the answer to a question is not explicitly stated in the source context paragraph. In other words, questions that require inference to ascertain implicit information are still a challenge. This is illustrated by several new datasets that require more sophisticated reasoning, like tracking state changes in processes (ProPara), and a host of other knowledge and reasoning types (ARC). An analysis on a subset of the ARC dataset suggests that a large proportion of questions (99/192, 52%) involve causal or physical knowledge (Boratko et al. 2018). An analysis by Crouse & Forbus (2016) suggests that 29% of the problems in 4th grade science tests require qualitative reasoning of the form QP theory provides.

The ProPara dataset (Dalvi et al. 2018) is the first large dataset of human generated natural language paragraphs about processes that are annotated with status, step, and location of participating entities. Along with the ProPara dataset, Dalvi et al. (2018) introduced two artificial neural network models to track state changes: a system that uses bilinear attention over sentences and an end-to-end system that uses bilinear attention over the entire paragraph. As of this writing, the two most successful models for ProPara enhance neural reading approaches with rules or knowledge graphs. Tandon et al. 2018 characterized ProPara as a structured prediction task, using commonsense rules derived VerbNet to avoid unlikely answers. Das et al. 2018 achieved state of the art results by recurrently building dynamic knowledge graphs that track entity locations. Das et al. 2018 also evaluated their system on a dataset of natural language recipes (Kiddon et al. 2018), which had previously been interpreted with neural process networks that simulate recipe actions and their effects (Bosselut et al. 2018). These recent papers suggest that commonsense knowledge and structured representations (e.g. in the form of knowledge graphs in Das et al. 2018 or domain-specific state predictors in Bosselut et al. 2018) are important for understanding the many complex aspects of procedural texts. We use ProPara to explore the step semantics framework and to understand how it can support some of these additional aspects of process understanding.

Step Semantics

Language is a blunt instrument. The challenge of learning by reading is to assemble, from both the signal in texts and the reader's preexisting knowledge, a reasonable extension of that reader's knowledge. Step semantics is a framework for specifying what a reader should learn from the language describing the steps of a process. Importantly, language enables people to intermingle continuous and discrete descriptions, hence our drawing together continuous processes, discrete actions, and events to provide the representational capacity necessary.

We call our account step semantics for two reasons. First, it is about the steps in a process viewed as a sequence of operations or events. (Operations, for recipes and procedures, events for natural processes that can be decomposed, such as life cycles and the formation of rain.) Second, often the internal structure of a step relies on one or more continuous processes, i.e. representable via the notion of process in qualitative process theory. At a coarser grain of description, the continuous changes are summarized via step changes (Rickel & Porter, 1994).

Ontology

We assume that a natural language description of a process consists of a sequence of sentences. The understanding process must create a description of states and steps. By state we mean an episode in a history (Hayes 1984), i.e. a set of propositional statements, including fluents, that is taken to hold over some time (instant or interval) describing a set of individuals. By step, we mean an event, or a set of events, that describes what happens during the transition between its before state and after state. The before/after relations impose a sequential ordering on states. This ordering can be cyclic, as in oscillations or life cycles. There can be alternate steps from a state, corresponding to events that either are alternatives to each other (e.g. bake in a microwave versus bake in an oven) or are occurring in parallel (e.g. the sprouting of legs and shrinking of its tale occurring at the same time in a tadpole's maturing).

The relationship between sentences and steps can be complicated. In the simplest case, e.g. ProPara, each sentence is assumed to be a single step, each state has at most one step leading to it and at most one step leading from it, and the order of events is given by the order of sentences. None of these assumptions hold more generally. The mapping between sentences and steps can be one to many. In the other direction, a step can be spread across multiple sentences in language. The incremental nature of natural language is why learning by reading systems using QP theory rely on a frame-based equivalent notation (McFate et al. 2014). In complex processes, e.g. recipes, steps can be undertaken in parallel (e.g. creating gravy while roasting a turkey), and can include multiple next steps (e.g. the reason to separate eggs is to do something different with the yolks versus the whites), and multiple previous steps (e.g. combining parts created by earlier steps). The temporal order in the events being different from the sequence of sentences describing them is very common in fiction, but is also used in instruction as a motivation. For instance, stories about why Hawaii caught a lucky break when Hurricane Lane dropped from a category 5 to a category 2 storm typically started with the good news and then described why this was such good news.

It should be clear from these complexities that understanding processes expressed in text, despite whatever progress is made on ProPara, remains a challenging problem.

Features

There are four fundamental kinds of steps:

- Changes of existence: A step can create or destroy something.
- Changes of property: A move step changes the location of something, for instance, and painting changes its color. Transformations, e.g. phase transformations such as boiling, change the type of an object.
- Change of quantity: A quantity change step indicates that the given quantity has risen or fallen during the step. The continuous processes that are causing this are often implicit. This is a useful thing to say if there are competing continuous processes occurring during a step, since knowing the result on a parameter of interest provides information about the relative magnitudes of effect. For example, evaporation from a bathtub is swamped by the change in mass from even a small stream of water flowing into it.
- Occurrence of a sub-process: A subprocess step describes the changes wrought by some process occurring within the larger process being described. For instance, if the water cycle is the process being described, there will typically be steps describing the roles of evaporation, condensation, and precipitation as part of that description.

These four types are mutually exclusive. As noted above, a single sentence may imply multiple steps, and a single step might be communicated by multiple sentences. A system with broad knowledge of the world will have representations encompassing multiple levels of detail and incorporating multiple perspectives (Falkenhainer & Forbus, 1991). This vocabulary of steps provides an interface layer between language and these representations, the specific level of detail and perspective depend on the level of detail in the natural language description. For instance, consider a moving object that is part of a larger mechanical system. Its movement might be simply described as a single change in property (i.e. location) step, or it may be described as a sub-step in the larger, more detailed description of the entire system.

Inertia is assumed for existence and property changes, i.e. if something exists then it continues to do so, until explicitly terminated or changed by some other step. Quantity changes, on the other hand, are subject the operations of continuous processes – one cannot melt chocolate, for example, and then leave it on the counter for an hour and assume it will remain molten.

Part of the hierarchy of process descriptions arises from hierarchies in place descriptions. In describing photosynthesis, for example, chloroplasts might be described as "in the leaf". A common heuristic is that the location of a process has to include the location of all of the constituents being used in it. So, the creation of sugar happens in the leaves, while the process as a whole must also include the roots and stems, since they collect and transport water that are used in the process.

When fluids are involved, we have found that both the classic piece of stuff and contained fluid ontologies (Hayes, 1984) are useful. In linear (cyclic or acyclic) steps, the moving liquid can be characterized in terms of molecular collections (Collins & Forbus, 1987), i.e. a specialization of the piece of stuff ontology such that the fluid moving is considered to be large enough to have macroscopic properties (e.g. temperature and pressure in moving water or air), but so small as to maintain coherence (e.g. not split at a fork in a piping system).

Connection to Language

We use FrameNet as a bridge between continuous processes and language (McFate & Forbus, 2016).

We note that there are many complexities in carving up constituent processes in language. For example, "Roots absorb water and minerals from the soil." Should this be viewed as two separate absorption processes, one for water and one for minerals? Without either additional knowledge or additional explanation, it is impossible to tell. Liquids are often used for transporting other things, in suspension or solution, in biological and engineered systems, and if the next sentence continues with "This combination of water and mineral flows…", then this expectation is satisfied. But in general there will be multiple possible interpretations which need to be maintained (or regenerated on backtracking) to understand such explanations. We begin by examining how simple steps can be recognized in terms of the verbs used in a sentence, then discuss how the semantics of verbs linked to processes can be used to extract additional information about a step.

Creation Steps: These are represented by the FrameNet frame `FN_Creating` and the linked Cyc event type `CreationEvent`. For biological creatures, the corresponding linked frames are `FN_Giving_birth` and `BirthEvent`. We note that FrameNet does not treat giving birth as a subframe of creating, but since Cyc does include `BirthEvent` as a specialization of `CreationEvent`, we treat this as a subcategory. The lexemes for this frame include create, assemble, form, formation, generate, make, produce, and several others.

Destruction Steps: These are represented by the FrameNet frame `FN_Destroying`, and the linked Cyc event type `DestructionEvent`. There are subframes for biological creatures, e.g. `FN_Killing`, `KillingByOrganism`.

Property Change Steps: There are quite a variety of these, e.g. FN_Cause_change, which can apply to names, religious beliefs, political climates, and so on. Similarly, FN_Change_of_phase_scenario covers phase changes such as freezing, boiling, and solidifying.

Quantity Change Steps: These include frames such as FN_Change_of_temperature, which covers verbs such as heat, warm, cool, chill, and refrigerate, and FN_Change_position_on_a_scale, which covers verbs such as rise, balloon, fluctuate, increase, etc.

Subprocess/Event Steps: Examples include FN_Motion, FN_Fluidic_motion, and FN_Giving. The role relations describe changes in the participants, e.g. prepositional phrases involving "from" and "to" identify the source and destination of something whose physical location or ownership changes.

Part of the complexity of natural language understanding of process descriptions comes from unpacking steps from the semantic interpretation. Another source of complexity is assembling a set of plausible temporal relationships among the steps. ProPara attempts to simplify these issues by treating each sentence as representing a single step, and assuming a strict identification of order of sentences with order of events that they describe. (An exception consists of cycles, where language like "continuing the cycle" indicates the existence of a cycle, but this lies outside the semantic representations stipulated in ProPara.)

Examples

To illustrate how step semantics can be used for natural language understanding, we use examples from the domains of recipes and ProPara.

Recipes

Consider the following recipe for French toast[1]:

1. In a small bowl, combine, cinnamon, nutmeg, and sugar and set aside briefly.
2. In a 10-inch or 12-inch skillet, melt butter over medium heat. Whisk together cinnamon mixture, eggs, milk, and vanilla and pour into a shallow container such as a pie plate. Dip bread in egg mixture. Fry slices until golden brown, then flip to cook the other side. Serve with syrup.

Despite being short and simple, this recipe includes several different types of steps (including some that are not explicitly stated) and is written in such a way that we cannot rely on steps being properly enumerated or being separated into distinct sentences. In the first step, the three dry ingredients (cinnamon, nutmeg, and sugar) undergo a mixing process which makes them individually irrecoverable. The second numbered step includes several actions (melting, whisking, pouring, dipping, frying, and flipping). While melting, butter undergoes changes in property (i.e. phase) and quantity (i.e. temperature). In whisking, individual wet and dry ingredients are combined and (in the same sentence) are poured into another container. In the bread-dipping step, ingredients are not destroyed, but there are changes in property (i.e. moisture and location). In the following sentences, multiple steps (frying, flipping) are combined into one sentence. In frying and flipping, there is a qualitative limit point ("until golden brown") and there is an implicit change of location before the final serving step.

Lexeme	*FrameNet Frames*	*Entities involved*
Combine	FN_Amalgamation, FN_Creation	Cinnamon, nutmeg, sugar
Melt	FN_Change_of_temperature, FN_Change_of_phase_scenario	Butter
Whisk	FN_Self_motion, FN_Amalgamation, FN_Creation	Cinnamon mixture, eggs, milk, vanilla
Dip	FN_Dunking	Bread, egg mixture
Fry	FN_Apply_Heat, FN_Change_of_temperature, FN_Amalgamation	Slices, (melted) butter
Flip	FN_Move_in_place	Bread

Table 1: Lexemes, frames, and entities for French toast recipe.

This recipe also exhibits a subtle temporal structure. Melting the butter does not need to happen before mixing ingredients and dipping bread, but all of those things need to happen before frying. These constraints cannot be inferred by the order that each of the steps is introduced, since each step does not necessarily depend on all steps previously mentioned. Instead, they can be inferred by reasoning about the entities involved in each step and their properties. Table 1 shows how individual lexemes and frames can be used to characterize each step.

The following recipe for roasted brussels sprouts[2] illustrates another complex temporal structure:

1. Preheat oven to 400 degrees F.
2. Cut off the brown ends of the Brussels sprouts and pull off any yellow outer leaves. Mix them in a bowl with the olive oil, salt and pepper. Pour them

[1] https://www.foodnetwork.com/recipes/robert-irvine/french-toast-recipe-1951408

[2] https://www.foodnetwork.com/recipes/ina-garten/roasted-brussels-sprouts-recipe2-1941953

on a sheet pan and roast for 35 to 40 minutes, until crisp on the outside and tender on the inside. Shake the pan from time to time to brown the sprouts evenly. Sprinkle with more kosher salt (I like these salty like French fries), and serve immediately.

Unlike the French toast recipe, this recipe describes steps such that each step necessarily begins before steps that are described later. However, the roasting step is supposed to temporally subsume the pan-shaking step (even though the recipe lacks a phrase like "while the sprouts roast…" to explicitly indicate that one step occurs during another). One way to make this inference is to identify the goal of pan shaking as a color property change of the sprouts (i.e. "browning") that is the ending condition for the roasting step.

These recipes are both relatively short and straightforward. However, they illustrate that (1) steps are not necessarily executed in the order that they are described, (2) that steps that are described with a single lexeme can denote multiple types of change (e.g. temperature and phase), and (3) that understanding the temporal constraints between steps hinges on the semantics of the processes (e.g. roasting, browning) and entities involved (e.g. pan).

ProPara

ProPara consists of 488 paragraphs about processes and a set of parameterized questions about the participants in each process paragraph. These questions concern when an entity is created, destroyed, or moved. Consider the following paragraph from the ProPara dataset:

"Chloroplasts in the leaf of the plant traps light from the sun. The roots absorb water and minerals from the soil. This combination of water and minerals flows from the stem into the leaf. Carbon dioxide enters the leaf. Light, water and minerals, and the carbon dioxide all mix together. This mixture forms sugar (glucose) which is what the plant eats. Oxygen goes out of the leaf through the stomata."

After reading this paragraph, a system ought to be able to answer questions like this one:

Q: Where is sugar produced?

A: In the leaf.

Our approach to answering these questions is to start with a general-purpose semantic parser, using a large knowledge base (NextKB[3]) and rich semantic interpretations based on Discourse Representation Theory (Kamp & Reyle, 2013), and use training data to customize the interpretation process for question answering. We call this *analogical Q/A training*. This approach has been used before on Geoquery (Crouse et al. 2018a), getting state of the art results with less data than typically required, and learning to recognize physical processes in paragraphs from science test questions (Crouse et al. 2018b). We combine this approach with step semantics to learn entailments from ProPara training data. The rest of this section describes how we do that and our preliminary results.

Analogical Q/A training works by taking natural language questions and some form of answers, and produces cases (i.e. sets of logical statements) that are retrieved and used in subsequent question answering. Typically natural language answers are provided, but here we use the table format provided by AI2, translated into predicate calculus, as shown in Figure 1. In training, the system is learning to map the FrameNet/Opencyc semantics it constructs to instances of events from the categories CreationEvent, DestructionEvent, and MovementEvent. We call it analogical Q/A training because what is created during the learning process are *query cases*, which are simple cases that provide a bridge between the logical forms produced by language and representations about processes. Queries to answer questions are generated by applying and composing query cases via analogy to interpret new texts.

(isa participant123 Participant)
(isa event123 CreationEvent)
(outputsCreated event123 participant123)
(outputsCreatedLocation event123 tolocation123)

(isa participant123 Participant)
(isa event123 DestructionEvent)
(inputsDestroyed event123 participant123)

(isa participant123 ProParaParticipant)
(isa fromlocation123 Location)
(isa tolocation123 Location)
(isa event123 MovementEvent)
(objectMoving event123 participant123)
(fromLocation event123 fromlocation123)
(toLocation event123 tolocation123)

Figure 1. Target logical form for each possible state change.

The process of constructing query cases during training works like this: First, the NLU system generates a set of syntactic and semantic choices, representing the space of possible interpretations for each sentence. Second, mappings between this space of interpretations and the target semantics (i.e. one of the three choices in Figure 1) are constructed. This involves using structural relations in the KB to find paths between concepts and relations. For example, here is a path that indicates that pulling is a kind of motion:

[3] NextKB integrates the OpenCyc ontology with FrameNet contents, a large lexicon, and support for qualitative and analogical reasoning. It will be available as a creative commons attribution resource shortly.

PullingAnObject → CumulativeEventType → Movement-Rotation → MovementEvent.

Role relations from the semantic interpretation are mapped to roles in the target logical form by using inheritance relations involving predicates, e.g.

objectActedOn → EventOrRoleConcept → objectMoving.

Typically there will be multiple potential matches, and these are filtered and scored based on constraints from Gentner's (1983) structure-mapping theory (e.g.1:1 mappings, prefer more systematic structures), albeit with re-representation occurring as part of the processing, similar to (Fan et al. 2009). The final step constructs query cases from these connections, and stores them into a case library for subsequent retrieval during Q/A.

Question-answering during testing proceeds as follows. Each test paragraph is read sentence by sentence. For each sentence, for each participant *p*, the following three categories of queries are asked: (Cat-1) Is *p* created (destroyed, moved) in the process? (Cat-2) When is *p* created (destroyed, moved)? (Cat-3) Where is *p* created (destroyed, moved from/to). These queries are processed by using analogical retrieval from the case library constructed during training. The best query cases are instantiated and ranked according to how well they match the sentence semantics. The consequents of the highest ranked query cases are used to predict the state change of the queried participant. Finally, all state changes are aggregated and the following common sense rules are applied to propagate the states of each participant:

1. Inertia: states are propagated, both forward and backwards, until a new state change occurs.
2. Collocation: If a participant X is converted to participant Y (X is destroyed when Y is created), and the position of Y is not known, then we assign the previous position of X to Y.

The combination of the queries and the common sense rules are used to generate a state change grid, in the format used by AI2, to compare against their answers. Table 2 compares our results on this task with the following models: ProComp (Clark et al. 2018), ProLocal, ProGlobal (both from Dalvi et al. 2018), ProStruct (Tandon et al. 2018), and KG-MRC (Das et al. 2018). Results are displayed as F1 scores for each category, as well as their respective macro-average. The ProStruct metric is different as the task was formulated as a structured prediction task.

While better than the prior rule-based model on two out of three categories, our approach does not yet out-perform the artificial neural network models, although it does better than ProLocal on two out of three categories, and better than all of them on Cat-2 questions. We believe there are at least two reasons for this. The first is the paucity of information extracted in cases currently, which does not provide enough discrimination during analogical retrieval, considerably reducing our recall score. We plan on exploiting more of the ontology and FrameNet information to address this. The second factor is that we were neither using coreference resolution nor the full set of commonsense rules used by the AI2 systems.

	Model	Cat-1	Cat-2	Cat-3	Macro averaged
Rule Based	ProComp	57.14	20.33	2.40	26.62
Artificial Neural Networks	ProLocal	62.65	30.50	10.35	34.50
	ProGlobal	62.95	36.39	35.90	45.08
	ProStruct	-	-	-	53.70*
	KG-MRC	62.86	40.00	38.23	47.03
Step Semantics	Our Model	49.50	43.92	17.13	36.85

Table 2. Comparison between models on ProPara dataset. Displayed values are F1 scores for each category, which are then macro-averaged. *ProStruct uses a different metric from previous papers.

Discussion

In this paper we propose a framework for representing state changes that occur in natural language descriptions of processes and procedures. Our analysis of recipes and learning experiment with process paragraphs suggests that this framework is capable of capturing some information from texts about processes.

We see several important lines of future work. First, we need to explore the ideas for improving ProPara performance noted above, to see how far we can push analogical Q/A training. It would not surprise us to find that exploiting additional linguistic and world knowledge during comprehension would lead to significant improvements. Second, we need to integrate step semantics into our learning by reading system, thereby enabling it to handle processes and procedures that go beyond ProPara, such as recipes including explicit cycles, forks, and joins, as well as moving beyond the 1:1 sentence/step model. These lines of work will, we hope, contribute to an account of human-level reasoning for question-answering about processes and procedures.

Acknowledgements

This research was supported in part by the Machine Learning, Reasoning, and Intelligence Program of the Office of Naval Research.

References

Allen, J.F. and Hayes, P.J (1990). Moments and Points in an Interval-Based Temporal Logic. Computational Intelligence, January.

Boratko, M., Padigela, H., Mikkilineni, D., Yuvraj, P., Das, R., McCallum, A., Chang, M., Fokoue-Nkoutche, A., Kapanipathi, P., Mattei, N., Musa, R., Talamadupula, K., and Witbrock, M. (2018). A systematic classification of knowledge, reasoning, and context within the ARC dataset. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 60–70. Association for Computational Linguistics.

Bosselut, A., Levy, O., Holtzman, A., Ennis, C., Fox, D., & Choi, Y. (2018). Simulating action dynamics with neural process networks. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chang, M.D. and Forbus, K.D. (2015). Towards Interpretation Strategies for Multimodal Instructional Analogies. *Proceedings of the 28th International Workshop on Qualitative Reasoning* (QR2015). Minneapolis, MN.

Chen, D., Fisch, A., Weston, J., & Bordes, A. (2017). Reading Wikipedia to Answer Open-Domain Questions. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., Tafjord, O. (2018). Think you have solved Question Answering? Try ARC, the AI2 Reasoning Challenge. arXiv:1803.05457v1.

Clark, P., Dalvi, B., & Tandon, N. (2018). What Happened? Leveraging VerbNet to Predict the Effects of Actions in Procedural Text. arXiv preprint arXiv:1804.05435.

Collins, J. and Forbus, K. (1987). Reasoning about fluids via molecular collections. Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-87), Seattle Washington (pp. 590-594).

Crouse, M. and Forbus, K. (2016). Elementary School Science as a Cognitive System Domain: How Much Qualitative Reasoning is Required? In *Proceedings of the Fourth Annual Conference on Advances in Cognitive Systems*. Evanston, IL.

Crouse, M., McFate, C.J., and Forbus, K.D. (2018a). Learning from Unannotated QA Pairs to Analogically Disambiguate and Answer Questions. *Proceedings of AAAI 2018*.

Crouse, M., McFate, CJ., & Forbus, K. (2018b) Learning to Build Qualitative Scenario Models from Natural Language. *Proceedings of QR 2018*, Stockholm.

Dalvi, B., Huang, L., Tandon, N., Yih, W. T., & Clark, P. (2018). Tracking State Changes in Procedural Text: a Challenge Dataset and Models for Process Paragraph Comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*(Vol. 1, pp. 1595-1604).

Das, R., Munkhdalai, T., Yuan, X., Trischler, A., & McCallum, A. (2018). Building Dynamic Knowledge Graphs from Text using Machine Reading Comprehension. arXiv preprint arXiv:1810.05682.

Drabble, B. (1993) EXCALIBUR: A program for planning and reasoning with processes. *Artificial Intelligence*, 62(1):1-40.

Fan, J., Barker, K., & Porter, B. (2009) Automatic interpretation of loosely encoded input. *Artificial Intelligence*, 173(2):197-220.

Forbus, K. (1984) Qualitative Process Theory. *Artificial Intelligence*, 24, pp. 85-168.

Forbus, K. (1989). Introducing actions into qualitative simulation. *Proceedings of IJCAI-89*.

Falkenhainer, B. and Forbus, K. (1991). Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51, 95-143.

Fikes, R. & Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189-208.

Friedman, S., Forbus, K., & Sherin, B. (2017). Representing, Running, and Revising Mental Models: A Computational Model. *Cognitive Science*, 42(4):1110-1145.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7, 155-170.

Hayes, P.J. (1984). The Second Naïve Physics Manifesto. In *Formal Theories of the Commonsense World.* Ablex, Norwood, NJ.

Hogge, J. (1987) Compiling plan operators from domains expressed in qualitative process theory. *Proceedings of AAAI-87.*

Kamp, H., & Reyle, U. (2013). *From discourse to logic: Introduction to model theoretic semantics of natural language, formal logic and discourse representation theory* (Vol. 42). Springer Science & Business Media.

Kiddon, C., Ponnuraj, G. T., Zettlemoyer, L., & Choi, Y. (2015). Mise en place: Unsupervised interpretation of instructional recipes. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 982-992)*.

McFate, C.J., Forbus, K. and Hinrichs, T. (2014). Using Narrative Function to Extract Qualitative Information from Natural Language Texts. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, Québec City, Québec, Canada.

McFate, C., and Forbus, K. (2016). An Analysis of Frame Semantics of Continuous Processes. *Proceedings of the 38th Annual Meeting of the Cognitive Science Society*, Philadelphia, PA, August.

McFate, C., and Forbus, K. (2016). Scaling up Linguistic Processing of Qualitative Process Interpretation. *Proceedings of the Fourth Annual Conference on Advances in Cognitive Systems*. Evanston, IL.

Rickel, J. & Porter, B. (1994) Automated modeling for answering prediction questions: Selecting the time scale and system boundary. *Proceedings of AAAI-94*

Ruppenhofer, J., Ellsworth, M., Petruck, M., Johnson, C., Baker, C., & Scheffczyk, J. (2016) *FrameNet 2: Extended Theory and Practice*. https://framenet2.icsi.berkeley.edu/docs/r1.7/book.pdf

Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2017). Bidirectional attention flow for machine comprehension. *Proceedings of the International Conference on Learning Representations (ICLR).*

Tandon, N., Dalvi, B., Grus, J., Yih, W. T., Bosselut, A., & Clark, P. (2018). Reasoning about Actions and State Changes by Injecting Commonsense Knowledge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 57-66).

Question Relatedness on Stack Overflow: The Task, Dataset, and Corpus-inspired Models

Amirreza Shirani[†], Bowen Xu[‡], David Lo[‡], Thamar Solorio[†] and Amin Alipour[†]
[†]University of Houston
[‡]Singapore Management University

Abstract

Domain-specific community question answering is becoming an integral part of professions. Finding related questions and answers in these communities can significantly improve the effectiveness and efficiency of information seeking. Stack Overflow is one of the most popular communities that is being used by millions of programmers. In this paper, we analyze the problem of predicting knowledge unit (question thread) relatedness in Stack Overflow. In particular, we formulate the question relatedness task as a multi-class classification problem with four degrees of relatedness.

We present a large-scale dataset with more than $300K$ pairs. To the best of our knowledge, this dataset is the largest domain-specific dataset for Question-Question relatedness. We present the steps that we took to collect, clean, process, and assure the quality of the dataset. The proposed dataset on Stack Overflow is a useful resource to develop novel solutions, specifically data-hungry neural network models, for the prediction of relatedness in technical community question-answering forums.

We adapt a neural network architecture and a traditional model for this task that effectively utilize information from different parts of knowledge units to compute the relatedness between them. These models can be used to benchmark novel models, as they perform well in our task and in a closely similar task.

Introduction

Community question answering (cQA) is becoming an integral part of professions allowing users to tap on crowds' wisdom and find answers to their questions. Techniques, such as answer summarization (Chan et al. 2012; Xu et al. 2017; Demner-Fushman and Lin 2006; Liu et al. 2008), question answer matching (Tan et al. 2016; Shen et al. 2015) and question semantic matching (Bogdanova et al. 2015; Wu, Zhang, and Huang 2011; Nakov et al. 2017), have been devised to improve users' experience by accelerating finding relevant information and enhancing the information presentation to users.

We refer to the collection of a question along with all its answers as a *knowledge unit* (KU). Finding related knowledge unit in these communities can significantly improve the effectiveness and efficiency of information seeking. It allows users to navigate between knowledge units, prune unrelated knowledge units from the information search space. Finding related knowledge units can be quite time-consuming due to the fact that even the same question can be rephrased in many different ways. Therefore automated techniques to identify related knowledge units are desirable.

In this work, we describe the task of prediction of *relatedness* in Stack Overflow, the most popular resource for topics related to software development. Knowledge in Stack Overflow is dispersed and developers usually need to explore several related knowledge units to gain insights into the problem at hand and possible solutions. Stack Overflow has become an indispensable tool for programmers; about 50 million developers visit it monthly, and over 85% of users visit Stack Overflow almost daily.[1] The reputation of this webiste has attracted many developers to actively participate and contribute to the forum. A study showed that most questions on Stack Overflow are answered within 11 minutes of posting them (Mamykina et al. 2011).

We formulate the problem of identification of related KUs, as a multi-class classification problem by breaking relatedness into multiple classes. More precisely, a model has to classify the degree of relatedness of two KUs into one of four classes: *duplicate*, *direct*, *indirect*, or *isolated*.

Predicting relatedness in Stack Overflow poses an interesting challenge because in addition to natural text, KUs contain a huge amount of programming terms which is of a different nature, and like many other cQA websites, different users exhibit different discursive habits in posting questions and answers; e.g., some provide minimal details in their questions or answers, while some tend to include a sizable amount of information.

We create a large, reliable dataset for training and testing models for this task. It contains more than 300K knowledge unit pairs annotated with their corresponding relatedness class. We report all steps to collect, clean, process, and assure the quality of the dataset. We rely on URL sharing in Stack Overflow to decide on the relatedness of KUs, as that programmers facing a specific problem are the best ones to judge the degree of relatedness of questions. We verified the

[1]Stack Overflow 2018 Developer Survey, `https://insights.stackoverflow.com/survey/2018/`

reliability of our approach by conducting a user study.

To establish a baseline for future evaluations, we present two successful neural network and traditional machine learning models. we adapt a lightweight Bidirectional Long Short-term Memory (BiLSTM) model tailored to our proposed dataset. We also investigate so-called soft-cosine similarity features in a Support Vector Machine (SVM) model. To investigate the adequacy of these models, we evaluate them on a closely related duplicate detection task. Our experiments show that our models outperform the state-of-the-art techniques in a duplicate detection task, suggesting that our models are potent benchmarks for our task.

Contributions. This paper makes the following contributions.

- We present the task of question relatedness in Stack Overflow, with four degrees of similarity.
- We present a reliable, large dataset for knowledge units relatedness in Stack Overflow.
- We adapt a corpus-inspired BiLSTM architecture for relatedness detection.
- We evaluate the performance of SVM models with several hand-crafted features to predict the relatedness in Stack Overflow.

Related Work

There are several tasks related to identifying semantically relevant questions such as Duplicate Question Detection (DQD), Question-Question similarity, and paraphrase identification.

Perhaps, one of the best-known general-domain DQD dataset is Quora [2] with more than 400K question pairs. Quora dataset was released on Kaggle competition platform in January 2017. Most of the questions on Quora are asked in one piece without any further description and are not restricted to any domain. Another well-known DQD dataset is AskUbuntu (Rodrigues et al. 2017). Similar to our Stack Overflow dataset, AskUbuntu dataset is acquired from Stack Exchange data dump [3] (September 2014). The differences are that AskUbuntu dataset only provides binary classes (DQD), it is 11 times smaller than our proposed dataset and only consist of titles and bodies in a concatenated form. Many solutions are proposed to address the DQD problem. (Bogdanova et al. 2015) utilized a convolutional neural network (CNN) to address the DQD problem on AskUbuntu and Meta datasets. (Silva et al. 2018) applied the same model on the cleaned version of datasets and showed that after removing Stack Exchange clues, the results drop by 20%. A more advanced architecture introduced in (Rodrigues et al. 2017) on AskUbuntu and Quora datasets. This model can be considered as the state-of-the-art model on AskUbuntu dataset which utilizes the combination of a *MayoNLP* model introduced in (Afzal, Wang, and Liu 2016) and a *CNN* model introduced in (Bogdanova et al. 2015). We use the same AskUbuntu dataset to evaluate our models on a secondary dataset. There are two major differences between our approach and the works in (Bogdanova et al. 2015) and (Rodrigues et al. 2017). First, we improve the performance of our model by computing the distance between title, body, and answers of the two knowledge units, whereas (Bogdanova et al. 2015) and (Rodrigues et al. 2017) only compute the similarity between title+body of the two knowledge units. Second, the hybrid architectures developed by (Rodrigues et al. 2017) is a complex CNN model along with 30k dense neural network followed by two hidden multilayers. However, our model uses shared layers bidirectional LSTMs with the limited number of parameters which results in a lightweight architecture.

Question-Question similarity introduced in subtask B of SemEval-2017 Task 3 on Community Question Answering [4] (Nakov et al. 2017) is one of the closest topics to our task. Although this task contains multi-classes of relatedness between two questions (i.e., PerfectMatch, Related, Irrelevant), unlike our task, the problem is formulated as a re-ranking Question_Question+Thread Similarity task. Various features were investigated to address Question-Question similarity introduced in subtask B of SemEval-2017 Task 3 such as neural embedding similarity features (Goyal 2017) and Kernel-based features (Filice, Da San Martino, and Moschitti 2017) (Galbraith, Pratap, and Shank 2017). The winner of this task is (Charlet and Damnati 2017) which utilized soft-cosine similarity features within a Logistic Regression model. Note that we employ the similar soft-cosine features in our traditional SVM model.

Duplicate detection between questions on Stack Overflow has been studied before. An approach named *DupPredictor* takes a new question as an input and tries to find potential duplicates of the question by considering multiple information sources (i.e., title, description and tags) (Zhang et al. 2015). *DupPredictor* computes the latent topics of each question by using a topic model. For each pair of questions, it computes four similarity scores by comparing their titles, descriptions, latent topics, and tags and then combined together to result in a new similarity score. In another similar work, (Xu et al. 2016) introduced a dataset for knowledge unit relatedness and proposed a convolutional neural network for predicting the relatedness. Unfortunately, the limited number of knowledge units (KUs) were collected heuristically and tend to have low quality. The presented dataset does not cover different parts of a knowledge unit, instead, it merges title+body into a single sequence. Clearly, mixing all parts together does not provide an opportunity to perform an experiment on separate parts of KUs independently. Moreover, this dataset contains some extra information (signals) which leads to a biased dataset. As explained in "Data Quality" section, we remove these unwanted clues from the data.

Description of The Dataset

Questions in the real world are supposed to have more relationships than only duplicate or non-duplicate. For example,

[2] https://goo.gl/kWCcD4

[3] https://askubuntu.com/

[4] http://alt.qcri.org/semeval2017/task3/

one question in Stack Overflow talks about *The time complexity of array function*[5], while another question is about *How to find time complexity of an algorithm*[6]. These two questions are linked by Stack Overflow users as related but not duplicate.

Relatedness Between Knowledge Units

Knowledge units often contain semantically-related knowledge, and thus they are linkable for different purposes, such as explaining certain concepts, approaches, background knowledge or describing a sub-step for solving a complex problem (Ye, Xing, and Kapre 2016). Figure 1 shows an example of how knowledge units are linked to each other on Stack Overflow. One of the answers of a knowledge unit (short for *KU1*) guides the asker to refer to another knowledge unit (short for *KU2*) which is helpful to solve the problem. These two knowledge units are linked through URL sharing. URL sharing is strongly encouraged by Stack Overflow to link related knowledge units (StackOverflow 2018). A network of linkable knowledge units constitutes a *knowledge unit network* (KUNet) over time through URL sharing (Ye, Xing, and Kapre 2016). Relationships between any two knowledge units in KUNet can be divided into four classes: *duplicate*, *direct*, *indirect* and *isolated* (Xu et al. 2016). Duplicate KUs discuss the same question and can be answered by the same answer. Direct relatedness between KUs means that the content of one KU can help solve the problem in the other KU, for example, by explaining certain concepts, providing examples, or covering a sub-step for solving a complex problem. Indirect relatedness means that contents of KUs are related but they are not immediately applicable to each other. Isolated KUs are not semantically related. The order of relatedness of each class is *duplicate* $>$ *direct* $>$ *indirect* $>$ *isolated*.

Dataset Creation

Figure 2 depicts the steps that we took to create a relatedness dataset. We describe each step below.

Extract preliminary data from Stack Overflow data dump. We mainly focus on Java-related knowledge units on Stack Overflow because Java is one of the top-3 most popular tags in Stack Overflow [7]. Moreover, questions with this tag not only are about Java programming language, but they cover a broad spectrum of topics that Java technology provides, such as web and mobile programming, and embedded systems. First, we extracted *all* knowledge units tagged by "Java" from Stack Overflow data dump. Next, all *duplicate* and *direct* links between knowledge unit pairs are extracted from Stack Overflow data dump.

Knowledge unit network. Knowledge unit network (KUNet) is a network in which each KU is represented as a node and an edge between two nodes exists if a *duplicate* or *direct* link exists between the two corresponding KUs. We construct a KUNet based on the extracted links from a table named *PostLinks* from Stack Overflow data dump.

[5] https://goo.gl/dJwmuE

[6] https://goo.gl/S81BjE

[7] https://stackoverflow. com/tags

Identifying duplicate and direct pairs As shown in Figure 2(a), the link between (A and B) and (B and C) are labeled as a *duplicate*. We also consider a *duplicate* link between A and C by transitivity. We apply transitivity rule until no new duplicate relation is found among knowledge units.

Identifying indirect and isolated pairs Four types linkable KU pairs are extracted from the KUNet based on their definitions. *Indirect* KU pairs are pairs of nodes that are indirectly connected in the network. More specifically, they are connected in the KUNet with a certain range of distance (in this case, length of shortest path $\in$ [2,5]), but the relationship between them belongs neither to duplicate nor direct. Finally, *isolated* KU pairs are pairs of nodes that are completely disconnected in the network.

Statistical Characteristics of the Dataset

Using the steps described in the previous section we created a dataset. Table 1 depicts the statistical characteristics of the dataset. The dataset contains 160,161 distinct knowledge units and 347,372 pairs of knowledge units with four types of relationships. Among all knowledge units, $117,139$ (i.e., 73%) of them have at least one code snippet in their body. The average number of words in code snippets in body is 118.46. There are $318,491$ answers in our dataset and each knowledge unit has 1.99 answers on average. $140,122$ (i.e., 87%) of knowledge units contain at least one answer and $90,672$ (i.e., 57%) of them contain one accepted answer. Moreover, $96,707$ (60%) of knowledge units have at least one code snippet in their answers which means that more than half of solutions are code related.

Training, Development, and Test Sets We split the dataset into three parts, train, development, and test, to facilitate the development, and evaluation of classification models. We assigned 60% of knowledge units to train set, 10% to development set, and 30% to test set. To have the same number of KU pairs for each class, by using under-sampling techniques, we make this dataset balanced.

Table 1: Brief statistics of the dataset

Scope	Indicator	Size
Whole KU	# of distinct KUs	160,161
	# of four types of KU pairs	347,372
Title	avg. # of words in title	8.52
Body	avg. # of words in body(exclude code snippets)	97.02
	# of distinct KUs whose body has at least one code snippet	117,139(73%)
	avg. # of code snippets in one body	1.46
	avg. # of words in single code snippet in one body	118.46
Answers	# of distinct answers	318,491
	avg. # of answers within single KU	1.99
	# of distinct KUs contain at least one answer	140,122(87%)
	# of distinct KUs contain an accepted answer	90,672(57%)
	# of distinct KUs whose answers has at least one code snippet	96,707(60%)
	avg. # of words in an answer (exclude code snippets)	68.39
	avg. # of code snippets within one answer	0.60
	avg. # of words in single code snippet	81.98

Instructions to Use The Dataset

Table 2 presents the overall structure of our dataset. There are 24 attributes in our dataset for each pair of knowledge units. The first 23rd attributes include all the content of the first and second knowledge units, they are id, title, body, accepted answer, answers, and tags.

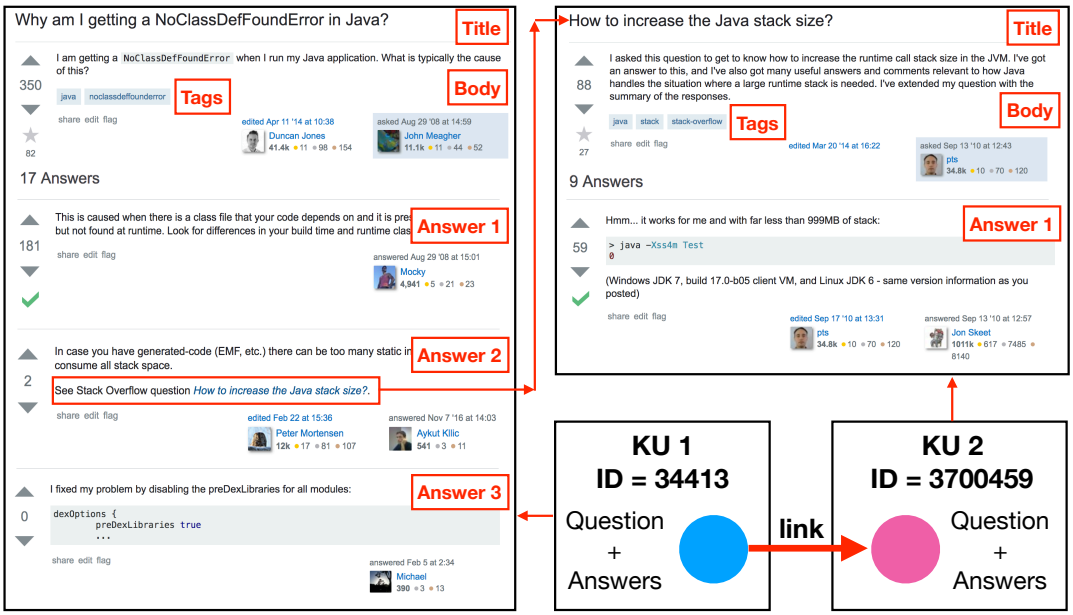
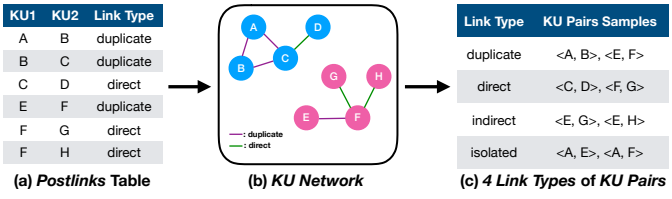


Figure 1: A pair of linkable knowledge units on Stack Overflow



(a) *Postlinks* Table **(b) *KU Network*** **(c) *4 Link Types* of *KU Pairs***

Figure 2: Overview of the data collection process

The last attribute (i.e., Attr. Id =24) represents the relationship between the two knowledge units (i.e., $< KU1, KU2, Relationship >$). More information is available at `https://anonymousaaai2019.github.io`

Table 2: The structure of the dataset

Attr. Id	Attr. Name	Attr. Description
1	Id	KU Pair ($< KU1, KU2 >$) Id
2/13	q1/2_Id	Id of *KU*'s Question on SO
3/14	q1/2_Title	*KU*'s Title
4/15	q1/2_Body	The text of *KU*'s Body (Exclude Code Snippets)
5/16	q1/2_BodyCode	Code Snippets in *KU*'s Body
6/17	q1/2_AcceptedAnswerId	Ids of *KU*'s Accepted Answers on SO
7/18	q1/2_AcceptedAnswerBody	The text of *KU*'s Accepted Answer (Exclude Code Snippets)
8/19	q1/2_AcceptedAnswerCode	Code Snippets in *KU*'s Accepted Answer
9/20	q1/2_AnswersIdList	Ids of *KU*'s Answers on SO
10/21	q1/2_AnswersBody	The text of *KU*'s Answers (Exclude Code Snippets)
11/22	q1/2_AnswersCode	Code Snippets in *KU*'s Answers
12/23	q1/2_Tags	Tags of *KU*
24	Class	Relationship (i.e., *duplicate*, *direct*, *indirect* or *isolated*)

Quality Control

Data Cleaning

We perform three operations to further improve the quality of our dataset. Natural language and programming language snippets are mixed in the text. To deal with this, first, we extract programming language snippets (aka. code snippets) from HTML formatted text by using the regular expression $\langle pre \rangle \langle code \rangle (.*?) \langle /code \rangle \langle /pre \rangle$ Note that, it is possible that multiple code snippets exist in body or multiple answers of one knowledge unit, so we store them into a list. Next, since text attributes (e.g., body, answer body) provided by Stack Overflow data dump are in HTML format, we clean the content by removing HTML tags and escape characters, e.g., $\langle p \rangle \langle /p \rangle$, $\&\#xA;$ and $\<$. Second, we observe and remove some extra information added by Stack Exchange API that can be considered as a signal. For example, at the beginning of the body content of some duplicate and direct questions, it includes the string `Possible Duplicate:`, followed by the topic content of the possible duplicate question. The inclusion of signals in training can result in a biased dataset and unreliable models. This problem was first observed by (Silva et al. 2018) in AskUbuntu dataset.

Third, we found that there is an overlap between some *duplicate* and *direct* links in the Stack Overflow data dump, since it provides knowledge unit pairs as long as two knowledge units are linked through URL sharing. To solve this, if a link belongs to *duplicate* and *direct* at the same time, we label it as a *duplicate*.

User Study

This dataset is extracted from Stack Overflow forum that is managed and maintained by volunteer domain experts who serve as moderators and contributors. Links between knowledge units (i.e., Stack Overflow posts) are validated in a crowdsourced process by domain experts. To asses the reliability of the crowdsourced process and our data collection procedure, we perform a user study. We ask three experts (who are not authors of this paper) to label relationships between pairs of knowledge units that we have in our dataset. The participants analyze a statistically significant sample

size (i.e., 96 pairs) that is representative of the population of knowledge units in our dataset (at 95% confidence level, and 10% margin or error). Each participant can provide his/her assessment of the degree of relatedness of two knowledge units in a 4 point Likert scale: 1 (unrelated/isolated), 2 (indirect), 3 (direct), and 4 (duplicate). The user study highlights that the participant labels are the same as the labels in our dataset 82% of the time. The average absolute difference between the Likert scores and the labels in our dataset is only 0.2 (out of 4). This highlights that the links in our dataset are of high-quality.

Method

In this section, we describe models to predict relatedness between knowledge units. We extensively explore different neural network and traditional models for this task and report the best-performing models. First, we investigate a BiLSTM architecture which progressively learns and compares the semantic representation of different parts of two knowledge units. The description of our model is presented in the next section. We then compare the BiLSTM model with a support vector machine model. We also apply these models to a closely similar task, duplicate detection in AskUbuntu, and compare the results with the state-of-the-art models in that task.

Data Pre-processing

We apply some simple pre-processing steps on all text parts, Title, Body and Answers. Since there are many technical terms in Stack Overflow, we apply more specific pre-processing steps: First, we split words with punctuation marks. For example, `javax.persistence.Query` `javax_query` changes to `javax persistence Query javax query`. Then, we split camel case words, for example, `EntityManage` is changed to `Entity Manage`. In the end, we take several standard steps in pre-processing data including: normalizing URLs and numbers, removing punctuation marks and stop-words, and changing all words to lowercase.

LSTM Model

We use bidirectional long short-term memory (BiLSTM) (Hochreiter and Schmidhuber 1997) as a sentence encoder to capture long-term dependencies in forward and backward directions. In a simple form, an LSTM unit contains a memory cell with self-connections, as well as three multiplicative gates to control information flow. Given input vector x_t, previous hidden outputs h_{t-1}, and previous cell state c_{t-1}, LSTM units operate as Figure 3, where i_t, f_t, o_t are input, forget, and output gates, respectively. The sigmoid function $\sigma()$ is a soft gate function controlling the amount of information flow. W_s and bs are model parameters to learn.

Figure 4 describes the overall architecture of the BiLSTM model (DOTBILSTM). Unlike previous studies (i.e. (Rodrigues et al. 2017)(Bogdanova et al. 2015)), this model utilizes the information in Title, Body and Answers parts of each knowledge unit. Each word (w_i) is represented as a vector, $\mathbf{w} \in \mathbb{R}^d$, looked up into an embedding matrix,

$$X = \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix}$$
$$i_t = \sigma(W_{iX}X + W_{ic}ct - 1 + b_i)$$
$$f_t = \sigma(W_{fX}X + W_{fc}ct - 1 + b_f)$$
$$o_t = \sigma(W_{oX}X + W_{oc}c_t - 1 + b_o)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{cX}X + b_c)$$
$$h_t = o_t \odot tanh(c_t)$$

Figure 3: LSTM Unit

$\mathbf{E} \in \mathbb{R}^{d \times |V|}$. A shared layer BiLSTM as a sentence encoder takes all the six inputs, embeds and transforms them into fixed-sized vectors. Then in order to compute the distance between each two knowledge units, we compute the inner dot product between all the three representations of the first knowledge unit and all three representations of the second knowledge unit. As a result, it maps a pair of knowledge units into a low dimensional space, where their distance is small if they are similar. In the next step, we concatenate computed values together. Our results show that concatenating the BiLSTM representations at the last layer increases the performance slightly. We feed these values to a fully-connected layer followed by a ReLU activation function, a dropout layer and then a *SoftMax* output layer for classification. The objective function is the Categorical cross-entropy objective over four class target labels.

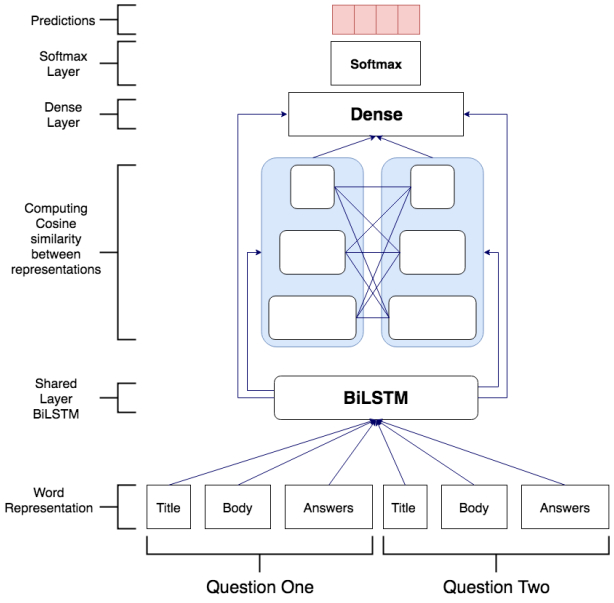


Figure 4: Main architecture of DOTBILSTM

Implementation Details (DOTBILSTM)

This section describes implementation details which are empirically chosen after running several models with different values and keeping the one that gives us the best results in the validation set.

We initialize word embeddings with pre-trained GloVe (Pennington, Socher, and Manning 2014) vectors of size 300. Compared to pre-trained Google news word2vec (Mikolov et al. 2013a) and word embedding trained on Stack Overflow, GloVe performed slightly better in this task. We choose the size of each sentence based on the average size over the training set. Titles are truncated or padded to 10 words, bodies to 60 words and answers to 180. BiLSTMs with 128 units is used as the encoder. In our experiments, we observed that using shared parameters for BiLSTMs boosts the model. The network uses Adam optimizer (Kingma and Ba 2014), and the learning rate is set to 0.001. The last layer is a dense layer with ReLu activation and 50 units. In order to have a better training and force the network to find different activation paths which leads to a better generalizing, a dropout layer with the rate of 0.2 is used. All the models are trained for 25 epochs and the reported test accuracy corresponds to the best accuracy obtained on the validation set.

SVM model

In this section, we explain the design of SOFTSVM, an SVM model for question relatedness task. We investigate different features as well as different data selections to achieve the best possible results.

We extract three types of features from knowledge units: *Number of common n-grams* which is simply the number of common word n-grams, and common character n-grams in a pair of text sequences. *Cosine similarity measure* to determine the similarity between two vectors (Kenter and De Rijke 2015; Levy, Goldberg, and Dagan 2015). This feature is obtained by TF-IDF weighting, computed over the training and development datasets. And, *Soft-cosine similarity measures* that unlike the traditional cosine similarity, takes into account word-level relations by computing a relation matrix (Sidorov et al. 2014). Given two N-dimension vectors a and b, the soft cosine similarity is calculated as follows.

$$soft-cosine(a,b) = \frac{\sum_{i,j}^{N} a_i m_{ij} b_j}{\sqrt{\sum_{i,j}^{N} a_i m_{ij} a_j}\sqrt{\sum_{i,j}^{N} b_i m_{ij} b_j}} \quad (1)$$

Unlike cosine similarity, soft-cosine similarity between two texts without any words in common is not null as soon as the two texts share related words. For computing the matrix M, we followed the same implementation presented in (Charlet and Damnati 2017), the winner of SemEval-2017 Task 3, Question-Question similarity. We create three variants of soft-cosine similarity feature. One is computed based on Levenshtein distance (Soft_Lev), and the other two features are based on two different word embeddings: Google News pre-trained word2vec (Mikolov et al. 2013a)(Soft_Google) and Stack Overflow domain-specific word2vec (Soft_SO).

Implementation Details (SOFTSVM)

We build an SVM model with the linear kernel using `sklearn` package (Pedregosa et al. 2011). In total, for each KU pair, we extract ten different hand-crafted features: three common word n-grams (for n=1,2 and 3), three common character n-grams (for n=3,4 and 5), cosine similarity and three soft-cosine similarity features Soft_SO, Soft_Lev and Soft_Google. We compute the features between titles, bodies and answers separately. For computing Soft_SO, we train word2vec on text parts of the dataset using skip-gram model (Mikolov et al. 2013b) with vectors dimension 200 and minimum word frequency of 20.

Feature Selection

In this section, we compare and select important features by building SVM models using each feature separately. As shown in Figure 5, cosine and three Soft-cosine features outperform other features. Therefore, we choose cosine similarity, Soft_SO, Soft_Google, and Soft_Lev, as the final feature set in the SOFTSVM because they perform better than other features.

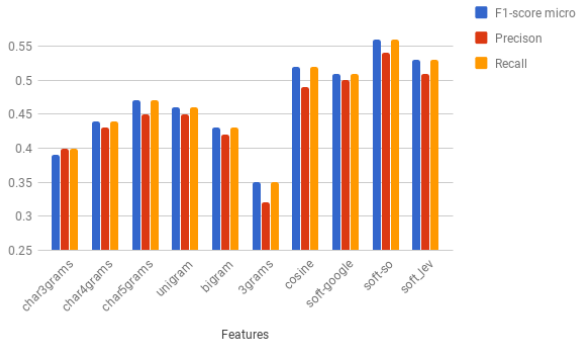


Figure 5: Performance of SVM models using individual features

To compare and select the important text selection parts, we build the SVM model by considering only title, body or answers. As shown in Table 3, the model with different parts perform similarly and the best performance is achieved when we consider all three, title, body and answers.

Table 3: Results of choosing different text selections.

Text selection/metrics	F-micro	Precision	Recall
Title	0.47	0.44	0.48
Body	0.51	0.49	0.51
Answers	0.51	0.5	0.51
Title, Body, Answers	0.59	0.58	0.59

Results and Discussion

Analysis of Results Table 4 compares results for both SOFTSVM and DOTBILSTM on Stack Overflow dataset. Comparing the obtained results, we realize that DOTBILSTM substantially outperforms SOFTSVM by more than 16 absolute percentage point in F-micro. This suggests that the BiLSTM model can utilize the large amount of training data in Stack Overflow dataset and predict the relatedness between knowledge units more effectively than our traditional model.

Table 4: Results for SOFTSVM and DOTBILSTM models

Model/Metrics	F-micro	Precision	Recall
SOFTSVM	0.59	0.58	0.59
DOTBILSTM	0.75	0.75	0.75

Tables 5 shows F-micro scores for predicting individual classes. Comparing results of the individual classes, DOTBILSTM performs better than SOFTSVM in predicting Isolated, Duplicate and Indirect classes.

Table 5: Comparing the results (f-score) of SOFTSVM and DOTBILSTM models

Models/Classes	Duplicate	Direct	Indirect	Isolated	Overall: Micro
SOFTSVM	0.53	0.57	0.44	0.79	0.59
DOTBILSTM	0.92	0.55	0.67	0.87	0.75

Reformulating the problem to the binary format of Duplicate Detection*:* For having a better comparison between our task and other typical duplicate/non-duplicate classification studies (some mentioned in "Related Work"), we *reformulate* the task to Duplicate Question Detection (DQD) and report the results of our models in the 2-class scenario. DQD is to predict if two given knowledge units are either duplicate or non-duplicate. To evaluate the models under the DQD scenario, we need to map four relatedness classes into two Duplicate and Not-duplicate classes. We consider duplicate class from the original dataset as *duplicate* and the rest as *non-duplicates* instances. To address the imbalanced class problem, we apply under-sampling techniques for non-duplicate class. More precisely, we randomly choose instances from all other three classes (direct, indirect and isolated) to have an equal number of both classes. By reformulating the task from multi-class to binary classification task, we expect our models to achieve higher results. We evaluate both DOTBILSTM and SOFTSVM models using the reformulated dataset. DOTBILSTM and SOFTSVM prediction performances increase to 0.91 and 0.70 f-score respectively. As we expected, by having two classes instead of four, in a relatively simpler problem, DOTBILSTM and SOFTSVM results increase by 16% percent and 11% respectively.

Comparing with AskUbuntu Dataset: We take a further step and expand our work by investigating AskUbutu DQD dataset for two specific reasons: (1) to show the robustness of the used models, and (2) To show the challenging nature of the proposed dataset on Stack Overflow compared to others. We expect to observe a different behavior of our models on this data due to the different nature and structure. For example, unlike Stack Overflow, the inputs of AskUbuntu dataset are only limited to title+body of each question. Moreover, AskUbuntu data contains a fewer number of instances, that is 24K pairs for training, 6K for testing and 1K for validation part. We use the cleaned version of AskUbuntu dataset (without signal) prepared by (Rodrigues et al. 2017). Using the same splitting used in (Rodrigues et al. 2017), our both models perform similarly. DOTBILSTM model achieves 0.88 f-score and 0.87 accuracy, and SOFTSVM model achieves 0.90 f-score and 0.90 accuracy. Our models outperform the state-of-the-art *Hybrid DCNN* model on this dataset introduced in (Rodrigues et al. 2017) with the accuracy of 0.79. This shows that not only our lightweight BiLSTM and traditional SVM model perform well in the Stack Overflow dataset, these models also outperform the complex *Hybrid DCNN* model on AskUbuntu dataset. Note that in order to evaluate the models on this dataset, we need to customize models to only have 2 inputs (title+body pairs).

More To Explore: In our experiments, we purposely confined our models to only utilize information in *Title*, *Body* and *Answers*. However, relying on other parts of the dataset like *BestAnswer*, *Tags* and *Code parts* can boost performance further for this task. As future work, we intend to investigate to utilize code parts as they are considered as informative resources about the content of the knowledge units.

Conclusion

This paper presents the task along with a large-scale dataset for identifying relatedness of knowledge unit (question thread) pairs in Stack Overflow. We reported all the steps for creating this dataset and a user study to evaluate the quality of the dataset. We devised two models, DOTBILSTM and SOFTSVM for this task and their performances for future evaluations. We also compared the performance of DOTBILSTM and SOFTSVM models with the state-of-the-art model on AskUbuntu dataset and found that these models outperform the state-of-the-art model. We made the dataset and models available online.

References

Afzal, N.; Wang, Y.; and Liu, H. 2016. Mayonlp at semeval-2016 task 1: Semantic textual similarity based on lexical semantic net and deep learning semantic model. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 674–679.

Bogdanova, D.; dos Santos, C.; Barbosa, L.; and Zadrozny, B. 2015. Detecting semantically equivalent questions in online user forums. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, 123–131.

Chan, W.; Zhou, X.; Wang, W.; and Chua, T.-S. 2012. Community answer summarization for multi-sentence question with group l 1 regularization. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, 582–591. Association for Computational Linguistics.

Charlet, D., and Damnati, G. 2017. Simbow at semeval-2017 task 3: Soft-cosine semantic similarity between questions for community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 315–319.

Demner-Fushman, D., and Lin, J. 2006. Answer extraction, semantic clustering, and extractive summarization for clinical question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the*

44th annual meeting of the Association for Computational Linguistics, 841–848. Association for Computational Linguistics.

Filice, S.; Da San Martino, G.; and Moschitti, A. 2017. Kelp at semeval-2017 task 3: Learning pairwise patterns in community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 326–333.

Galbraith, B.; Pratap, B.; and Shank, D. 2017. Talla at semeval-2017 task 3: Identifying similar questions through paraphrase detection. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 375–379.

Goyal, N. 2017. Learningtoquestion at semeval 2017 task 3: Ranking similar questions by learning to rank using rich features. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 310–314.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kenter, T., and De Rijke, M. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, 1411–1420. ACM.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Levy, O.; Goldberg, Y.; and Dagan, I. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225.

Liu, Y.; Li, S.; Cao, Y.; Lin, C.-Y.; Han, D.; and Yu, Y. 2008. Understanding and summarizing answers in community-based question answering services. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, 497–504. Association for Computational Linguistics.

Mamykina, L.; Manoim, B.; Mittal, M.; Hripcsak, G.; and Hartmann, B. 2011. Design lessons from the fastest q&a site in the west. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, 2857–2866. New York, NY, USA: ACM.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Nakov, P.; Hoogeveen, D.; Màrquez, L.; Moschitti, A.; Mubarak, H.; Baldwin, T.; and Verspoor, K. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 27–48.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research* 12(Oct):2825–2830.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Rodrigues, J. A.; Saedi, C.; Maraev, V.; Silva, J.; and Branco, A. 2017. Ways of asking and replying in duplicate question detection. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, 262–270.

Shen, Y.; Rong, W.; Sun, Z.; Ouyang, Y.; and Xiong, Z. 2015. Question/answer matching for cqa system via combining lexical and sequential information. In *AAAI*, 275–281.

Sidorov, G.; Gelbukh, A.; Gómez-Adorno, H.; and Pinto, D. 2014. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas* 18(3):491–504.

Silva, J.; Rodrigues, J.; Maraev, V.; Saedi, C.; and Branco, A. 2018. A 20% jump in duplicate question detection accuracy? replicating ibm teams experiment and finding problems in its data preparation. *META* 20(4k):1k.

StackOverflow. 2018. How to ask a good question?, `http://stackoverflow.com/help/how-to-ask`.

Tan, M.; dos Santos, C.; Xiang, B.; and Zhou, B. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 464–473.

Wu, Y.; Zhang, Q.; and Huang, X. 2011. Efficient near-duplicate detection for q&a forum. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, 1001–1009.

Xu, B.; Ye, D.; Xing, Z.; Xia, X.; Chen, G.; and Li, S. 2016. Predicting semantically linkable knowledge in developer online forums via convolutional neural network. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 51–62. ACM.

Xu, B.; Xing, Z.; Xia, X.; and Lo, D. 2017. Answerbot: automated generation of answer summary to developersź technical questions. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, 706–716. IEEE Press.

Ye, D.; Xing, Z.; and Kapre, N. 2016. The structure and dynamics of knowledge network in domain-specific q&a sites: a case study of stack overflow. *Empirical Software Engineering*.

Zhang, Y.; Lo, D.; Xia, X.; and Sun, J.-L. 2015. Multi-factor duplicate question detection in stack overflow. *Journal of Computer Science and Technology* 30(5):981–997.