Complex Question Decomposition for Semantic Parsing

Haoyu Zhang[†], **Jingjing Cai**[‡], **Jianjun Xu**[†], **Ji Wang**[†]
[†]HPCL, College of Computer, National University of Defense Technology, China
[‡]Meituan-Dianping Group, China
{zhanghaoyu10, jjxu, wj}@nudt.edu.cn
caijingjing02@meituan.com

Abstract

In this work, we focus on complex question semantic parsing and propose a novel Hierarchical Semantic Parsing (HSP) method, which utilizes the decompositionality of complex questions for semantic parsing. Our model is designed within a three-stage parsing architecture based on the idea of decomposition-integration. In the first stage, we propose a question decomposer which decomposes a complex question into a sequence of sub-questions. In the second stage, we design an information extractor to derive the type and predicate information of these questions. In the last stage, we integrate the generated information from previous stages and generate a logical form for the complex question. We conduct experiments on COMPLEXWEBQUESTIONS which is a large scale complex question semantic parsing dataset, results show that our model achieves significant improvement compared to state-of-the-art methods.

1 Introduction

Semantic parsing is a task which maps natural language utterances into logical forms such as SQL queries that can be executed based on relational databases or knowledge bases directly. Semantic parsing is a long-standing and difficult problem in natural language processing. In recent studies, researchers usually treat natural language descriptions/questions as input and use different sequence-to-sequence frameworks to generate logical forms (Xu et al., 2017; Dong and Lapata, 2016). However, these methods ignore the decompositionality of a complex question which is usually composed of a set of sub-questions, the understanding of each sub-question could contribute to the semantic parsing of the original complex question.

Figure 1 gives an example of a complex question and its logical form. The related sub-questions in stage-1 and the corresponding predicate (relation) information of each sub-question in stage-2 could help to obtain the logical form of the complex question in stage-3.

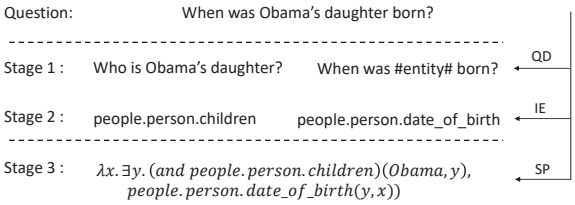

Figure 1: Example of question decomposition(*QD*), information extraction(*IE*) and semantic parsing(*SP*).

Question decomposition is important and many previous work utilize the decompositionality of complex questions to help question understanding. Kalyanpur et al. (2012) propose to use a suite of decomposition rules for question decomposition. The drawback of rule-based methods is that it needs experts to design rules and the rules are usually with low coverage and hard to be extended to other domains and tasks. Talmor and Berant (2018) propose a neural question decomposition approach to answer complex questions. They use the pointer network (Vinyals et al., 2015) to generate splitting points in the complex question and separate the complex question into a sequence of simple questions. This neural-based method alleviates the cost of manually designed rules or features. However, sometimes decomposing a complex question by splitting points may not find best sub-questions, and thus lose some information. For example, the sub-question "Who is Obama's daughter?' can not be generated by the splitting points of the complex question in Figure 1. To address the above problem, we propose to use a more flexible neural generative question decom-

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4477–4486
Florence, Italy, July 28 - August 2, 2019. ©2019 Association for Computational Linguistics

poser to directly generate complete and natural sub-questions based on an input complex question, without word order and content restrictions.

To parse a complex question into its corresponding logical form, we propose a hierarchical semantic parsing (HSP) model which is designed as a hierarchical neural sequence-to-sequence architecture. The underline idea of our HSP model is decomposition and integration. Specifically, as shown in Figure 1, our HSP model first decomposes a complex question into sub-question sequence with a question decomposer (QD), and then extracts key semantic information based on the generated sub-questions and the original complex question with an information extractor (IE). Finally, HSP model integrates the previously generated auxiliary information and generates the logical form of the complex question. Our HSP model can be seen as a multi-stage reasoning process, with each stage focusing on different level of information and reducing the search space of logical forms step-by-step by integrating previously generated information.

The main contributions of this paper are threefold:

1. We propose an effective and flexible question decomposition method;

2. We propose a hierarchical semantic parsing model based on a sequence-to-sequence paradigm which incorporates a question decomposer and an information extractor;

3. Experimental results demonstrate that the proposed model achieves a significant improvement in semantic parsing performance.

2 Related Work

2.1 Semantic Parsing

Typically, traditional semantic parsing models (Zettlemoyer and Collins, 2005; Mooney, 2007; Liang et al., 2011; Cai and Yates, 2013; Artzi et al., 2013; Kwiatkowski et al., 2013; Berant et al., 2014; Yih et al., 2015; Yao, 2015) are learned based on carefully designed features. For instance, Kwiatkowski et al. (2011) propose a combinatory categorical grammar induction technique for semantic parsing with different levels of features. Liang et al. (2011); Reddy et al. (2014) build semantic parsers without relying on logical form annotations but through distant supervision. Xiao et al. (2016); Yin and Neubig (2017) use syntax information to improve semantic parsing models. Fan et al. (2017) apply a transfer learning method in semantic parsing. To alleviate the cost of feature engineering, neural semantic parsing approaches have attracted significant attention (Jia and Liang, 2016; Dong and Lapata, 2016; Herzig and Berant, 2017; Gardner et al., 2018; Goldman et al., 2018; Chen et al., 2018; Dong et al., 2018; Dong and Lapata, 2018). For example, Jia and Liang (2016) propose a framework to introduce data recombination and train a sequence-to-sequence model for semantic parsing. Dong and Lapata (2018) propose to firstly parse a question to a coarse logical form then a fine-grained one based on a neural architecture. However, these approaches miss the opportunity to utilize question decomposition information for complex question semantic parsing. In this work, we leverage a sequence-to-sequence architecture and design a neural hierarchical sequence-to-sequence model to capture the syntactic structure, e.g., question decomposition information of complex questions.

2.2 Question Decomposition

Question decomposition has been successfully used in complex question answering (Kalyanpur et al., 2012; Iyyer et al., 2016; Talmor and Berant, 2018; Song et al., 2018). Kalyanpur et al. (2012) propose a framework using decomposition rules to identify facts in complex questions based on lexicon-syntactic features. The model then leverages the identified facts alone with a question rewriting component and a candidate reranker to generate final ranked answer list. Their work rely on feature engineering and manually designed rules which is difficult to be adapted to applications in other domains. Iyyer et al. (2016) propose a method for complex question answering based on tables. To answer complex questions, they split each complex question into several inter-related simple questions by crowd-sourcing, and design an end-to-end neural model to predict the answer based on the simple questions. Talmor and Berant (2018) propose a splitting-based question decomposition model to find splitting points in the original complex question and decompose it into a sequence of sub-questions. They then use a machine reading comprehension method to get the answers of each sub-question and compose the answers to obtain answer of the complex

Question	When was Obama's daughter born?
DR	Who is Obama's daughter? # When was **#entity#** born?
SR	composition # people.person.children # people.person.date_of_birth

Table 1: Example of intermediate representations, **#entity#** in one sub-question represents a placeholder, and the real word is filled in by the answer of another sub-question.

question. Sub-questions obtained by this method are usually incomplete. In this work, we propose a neural generative question decomposition approach to directly generate complete and natural sub-questions, which also improves the performance of complex question semantic parsing.

3 Model

In this section, we introduce the architecture of our hierarchical semantic parsing model. The model receives complex question inputs and generates logical forms. It combines the sequence-to-sequence paradigm with a hierarchical parsing mechanism in a differentiable way and can be trained end-to-end.

3.1 Model Overview

Our model treats complex questions and logical forms as sequences, learns to generate logical forms for questions. We denote a complex question as $x = \{x_1, \cdots, x_{|x|}\}$, and logical form as $y = \{y_1, \cdots, y_{|y|}\}$.

To better model and generate logical forms, our model utilizes two types of intermediate representations: the decomposed representation(*DR*) and the semantic representation(*SR*). *DR* consists of decomposed simple questions and *SR* contains key information of the original complex question including question type and all predicates in the question. An example of the two intermediate representations format is shown in Table 1. Decomposed representation is denoted as $z = \{z_1, \cdots, z_{|z|}\}$ and semantic representation is denoted as $w = \{w_1, \cdots, w_{|w|}\}$. Each training sample is a $< x, y, z, w >$ quad.

3.2 Basic Architecture

First we illustrate the basic structure of our model: a parsing unit. A parsing unit consists of an encoder network and a decoder network, based on the multi-head attention encoder/decoder of Transformer (Vaswani et al., 2017). Its input has two parts: the input sequence and additional information, and the output is the parsed target sequence. The input sequence and target sequence are text utterances, and additional information is a sequence of vectors representing encoding for some kind of auxiliary information.

In this subsection, we represent the input sequence of the paring unit as $a = \{a_1, \cdots, a_{|a|}\}$, input additional information as $e = \{e_1, \cdots, e_{|e|}\}, e_i \in R^n$, and output sequence as $o = \{o_1, \cdots, o_{|o|}\}$.

3.2.1 Encoder

On the encoder side, the parsing unit encodes the input sequence a to context aware representation $h = \{h_1, \cdots, h_{|a|}\}, h_i \in R^m$. We introduce the Transformer encoder (Vaswani et al., 2017) here.

The encoder first maps the sequence to word representations and then generates the output using a L layer Transformer encoder. The total process is denoted by:

$$h = f_{enc}(a) = f_{enc}^{proc}(f_{enc}^{emb}(a)) \quad (1)$$

3.2.2 Decoder

The decoder receives encoder output h and input additional information e, first fuses the two encoded representations by concatenating them to get fused representation $[h, e]$.

At decoder time step t, with fused representation $[h, e]$ and previous decoded output $o_{<t} = \{o_1, \cdots, o_{t-1}\}$, decoder calculates conditional probability $P(o_t|o_{<t}, [h, e])$.

First decoder embedding function f_{dec}^{emb} maps previous decoder outputs $o_{<t}$ to word embeddings and add positional encoding to get decoder word representations. Like the encoder, decoder also stacks L identical layer and the word representations are then fed to these layers along with fused representation $[h, e]$. If we represent the l-th layer output vector of position j as k_j^l and represent l-th layer previous output as $k_{\leq j}^l = \{k_1^l, \cdots, k_j^l\}$, the decoder layer output is $k_j^l = Layer(k_{\leq j}^{l-1}, [h, e])$.

Given the last layer output k_j^L, the probability of current word $P_{vocab}^j(w)$ on target vocabulary

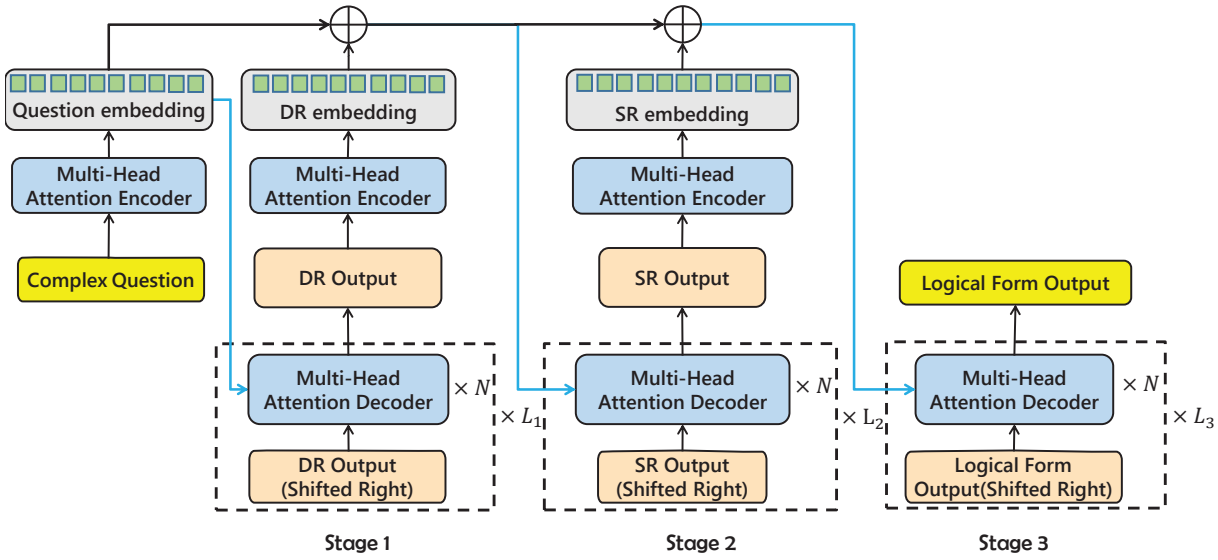

Figure 2: Model Overview, L_1, L_2, L_3 represent length of the corresponding decoder's output, and N represents the decoder layer number. Yellow rectangles denote input and output sequence, orange rectangles denote intermediate output utterances, and gray ones are encoded representations. *DR* and *SR* represent decomposed representation and semantic representation respectively.

$\mathcal{V} = \{w_1, \cdots, w_{|v|}\}$ is calculated as the following equation, where W_o, b_o is parameters:

$$P^j_{vocab}(w) = Softmax(W_o \cdot a^L_j + b_o) \quad (2)$$

The decode process is triggered with the start of sequence token "[BOS]" and terminated on the end of sequence token "[EOS]".

3.2.3 Copy Mechanism

To tackle out-of-vocabulary words, we incorporate copy mechanism (Gu et al., 2016) in the decoder.

At decode time step t, first we calculate the attention distribution over source sequence a using the bilinear dot product of last layer decoder output k^L_t and encoder output h, as Eq. 3 4 shows.

$$u^i_t = k^L_t W_q h_i \quad (3)$$
$$\alpha_t = Softmax(u_t) \quad (4)$$

Then we calculate copy probability $P^t_{copy} \in [0, 1]$ as following equation. W_q, W_g, b_g are learnable parameters:

$$P^t_{copy} = \sigma(W_g \cdot [k^L_t, h, e] + b_g) \quad (5)$$

Using P^t_{copy} we calculate the weighted sum of copy probability and generation probability to get the final predicted probability of extended vocabulary $\mathcal{V} + \mathcal{X}$, where $\mathcal{X}$ is set of out of vocabulary words in source sequence a:

$$P_t(w) = (1 - P^t_{copy})P_{vocab}(w) + P^t_{copy} \sum_{i:w_i=w} \alpha^i_t \quad (6)$$

The decoding process is formulated by Eq. 7. Note here that we use f^t_{dec} to represent one time step of the decoder with the copy mechanism process. For brevity we roll all time steps of the decoder, using Eq. 8 to denote $P(b|[h, e])$.

$$o_t = f^t_{dec}(f^{emb}_{dec}(o_{<t}), [h, e]) \quad (7)$$
$$o = f_{dec}([h, e]) \quad (8)$$

Following is the loss function of the basic architecture with parameters θ, b^*_t is the target word in time step t:

$$\mathcal{L}(\theta) = \frac{1}{T} \sum_{t=1}^{T} -\log P(o_t = o^*_t | a, e, o_{<t}) \quad (9)$$

3.3 Hierarchical Semantic Parsing

We now introduce HSP based on the above basic architecture, HSP is a bottom up multi-stage parsing process. Figure 2 illustrates the typical three stage HSP structure of our model, each stage process is similar to the basic architecture we elaborate above.

As illustrated in the model overview subsection, we denote the input question as x and the

output logical form as y. The train objective of the basic architecture in Eq. 9 directly minimizes the cross entropy between conditional probability $P(y|x)$ and true probability of target sequence $P(y^*)$. HSP mechanism turns the process into a multi-stage process by splitting the objective to several conditional probabilities' products. For our three stage HSP model shown in Figure 2, the objective is $P(y|x,z,w)P(w|x,z)P(z|x)$, in which z and w represent decomposed representation and semantic representation respectively.

3.3.1 Question Decomposer

On the first stage of HSP, we design a question decomposer to decompose the complex question to simple question sequences. The input of the question decomposer is the complex question x, and the output is the decomposed representation z. The model first maps the input x to context aware representations h using the question encoder $h = f_{enc_1}(x)$, at this stage no additional information is given, so fused representation is identical to h. Then with a decomposed decoder, the decomposed representation is predicted: $z = f_{dec_1}(h)$. In Figure 2 the decoding process is unrolled to time steps and surrounded by a dotted frame, at each time step previous outputs are shifted right and fed into the decoder. The beginning of the blue line pointing to the decoder is fused representation used by the decoder, for question decomposer it is the question embedding.

3.3.2 Information Extractor

The second stage of HSP extracts key information of complex questions, from the complex question itself and the decomposed simple questions. The input sequence of the information extractor is decomposed representation, additional information is question embedding, and the target output sequence is semantic representation. The encoder process encodes decomposed representation z using sub-question encoder: $h^z = f_{enc_2}(z)$. The fused representation $[h, h^z]$ is then fed into the semantic decoder to decode semantic representation: $w = f_{dec_2}([h, h^z])$. In Figure 2, the $\oplus$ notation on the top denotes the representation fusing process.

3.3.3 Semantic Parser

The final stage of the HSP model is a semantic parser. It receives the context aware embedding of complex question and decomposed representation, and semantic representation sequence. It encodes the semantic representation $h^w = f_{enc_3}(w)$, concatenates the three part of representation $[h, h^w, h^z]$, and logical form are predicted conditioned upon the fusing representation: $y = f_{dec_3}([h, h^w, h^z])$.

While the loss function of the basic architecture is as shown in Eq. 9, the training objective of HSP model is to minimize following loss functions as Eq. 10, where $L_1 = -\log P(z|x)$, $L_2 = -\log P(w|x,z)$ and $L_3 = -\log P(y|x,z,w)$ denotes losses of three stages. λ_1, λ_2 in the equation are two hyperparameters.

$$\mathcal{L}_{HSP}(\theta) = \lambda_1 \cdot L_1 + \lambda_2 \cdot L_2 + L_3 \qquad (10)$$

During inference, the model uses a three stage inference process, first getting the prediction of decomposed representation $\hat{z} = argmax_z P(z|x)$, and then predicting semantic representation $\hat{w} = argmax_w P(w|x,z)$, finally predicting logical form $\hat{y} = argmax_y P(y|x,z,w)$. Each sequence is obtained using a greedy search method like beam search.

From a cognitive view, HSP can be seen as another form of attention mechanism, it helps the model concentrate on the most important semantic part first, and fills other skeletons step by step. From the point of modeling, HSP simplifies the generation by splitting the semantic part with logical form grammars, which simplifies the modeling task of each process. The HSP mechanism can also be regarded as a kind of information flow, the information parsed on the previous stages can provide a soft constraint for the generation process at a later stage.

Note that we just introduce one particular form of HSP for semantic parsing in this section. HSP is actually a mechanism that is highly flexible; its structure can be applied to any sequence-to-sequence framework and used in many structured sequence generation tasks.

4 Experiment

4.1 Settings

During our experiments, we build a vocabulary for complex questions, all intermediate representations and logical forms. The vocabulary contains up to 30K words, constructed from all words with more than 4 occurrences in the corpus. All out-of-vocabulary words are represented by UNK.

Our model uses pre-trained 6B tokens 300 dimensional Glove word embeddings (Pennington et al., 2014), for vocabulary words which do not have pre-trained embeddings(including three special words: UNK, BOS and EOS), we assign them uniform randomized values. During training, we update all word embeddings.

Our model also uses a pre-trained Stanford-CoreNLP POS model (Manning et al., 2014) in the encoder embedding process. We use categorical POS annotations and map them to POS embedding vectors of dimension 30, the POS embedding vectors are initialized from uniform distribution $\mathcal{U}(-0.1, 0.1)$ and updated during training. The POS embeddings are concatenated with word embeddings to generate word representations.

We fix hidden size of all encoder and decoder units to 300. The encoder and decoder of all HSP models are stacked by 6 identical layers. We train the model using Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$ and use dynamic learning rate during training process. For regularization, we use dropout (Srivastava et al., 2014) and label smoothing (Szegedy et al., 2016) in our models and set the dropout rate to 0.2, set the label smoothing value to 0.1.

During training, we train our models using minibatches of 128 samples, all models are trained for at most 20,000 steps, selecting the best model based on development set performance. After one model is trained, we use beam search of beam size 16 to generate logical form sequences.

The implementations of our model would be released for further study[1].

4.2 Dataset

To evaluate the performance of our model on semantic parsing, we conduct experiments on ComplexWebQuestions(v1.0) dataset (Talmor and Berant, 2018) released here[2], which is built on the WebQuestions dataset (Berant et al., 2014) and consists of samples of complex question, decomposed question sequence and sparql format logical form. ComplexWebQuestions is a large scale semantic parsing dataset and contains 27734 training samples, 3480 development and 3475 test samples.

The dataset has four types of complex questions: composition (46.7%), conjunctions (42.4%), superlatives (5.3%) and comparatives (5.6%). Each question is either the combination of two simple questions, or an extension of a simple question. We identify entities in logical forms and replace them with placeholders during training and inference.

4.3 Results

We measure model performance by calculating the accuracy of generated logical forms, and compare performance of our approach(HSP) with various competitive baselines. In table 2, SP Unit denote for the semantic parsing unit, it uses the basic structure of HSP model with no intermediate representations, cooperates POS embedding, copy mechanism and Glove word embedding together with the Transformer.

Table 2 presents all models' accuracy on development and test set. Note that we treat SP Unit as the performance baseline and calculate other models' accuracy gain or decline compared to it, recorded in parentheses in the table. SP Unit gets 59.91% accuracy on test set, 8.91% higher than Pointer-Generator which matches 51% golden sparql queries. We also observe that the performances of SEQ2SEQ and SEQ2TREE are lower than Pointer-Generator, the two models get 47.3% and 49.68% accuracy on test set. We think the reason is that Pointer-Generator's copy mechanism helps logical form generation. Transformer achieves 53.41% on the test set which is also 6.5% lower than SP Unit but higher than Pointer-Generator. This group of experiment proves that semantic parsing on ComplexWebQuestions is difficult for traditional sequence-to-sequence models, and SP Unit is more effective than some previous systems. The reason is that by combining self-attention with copy mechanism, POS embedding and other modules, SP Unit has good modeling ability for logical forms of complex questions.

Coarse2Fine obtains 53.52% accuracy on the test set which is 1.84% lower than SP Unit. Our HSP model outperforms SP Unit by 6.27% accuracy which is a wide margin (with SP Unit as a baseline, the relative improvement of HSP is 10.5%). It proves the effectiveness of HSP mechanism. Compared to other neural semantic parsing models, HSP achieves significant improvement, proving that incorporate sub-questions and key information together boost logical form generation effectively. We think the key reason is that ques-

[1] https://github.com/cairohy/hsp
[2] https://www.tau-nlp.org/compwebq

Model	Dev (%)	Relative_perf	Test (%)	Relative_perf
SEQ2SEQ (Dong and Lapata, 2016)	50.22	-11.47	47.30	-12.61
SEQ2TREE (Dong and Lapata, 2016)	51.87	-9.82	49.68	-10.23
PointerGenerator (See et al., 2017)	53.10	-8.59	51.00	-8.91
Transformer (Vaswani et al., 2017)	56.78	-4.91	53.41	-6.50
Coarse2Fine (Dong and Lapata, 2018)	58.59	-3.10	58.07	-1.84
SP Unit	61.69	/	59.91	/
HSP w/o DR	66.09	+4.40	63.16	+3.25
HSP w/o SR	67.32	+5.63	64.48	+4.57
HSP(Switch)	68.13	+6.44	65.29	+5.38
HSP	**68.79**	**+7.10**	**66.18**	**+6.27**

Table 2: Logical form accuracy on development and test set of ComplexWebQuestions dataset. The second column and forth column of the table show relative performance compared to the SP Unit on dev set and test set separately.

tion decomposition turns the complex question into simple questions and then solves simple questions in a divide-and-conquer manner, which simplify representation learning process of the model in each stage.

4.3.1 Ablation Analysis

As the above part of Table 2 shows, we conduct an ablation study on HSP to analyze the importance of each component in HSP. We use four HSP ablation models for experiments. HSP(Switch): A three stage model which switches the order of intermediate parsing targets, first parsing semantic representation and then decomposing the original question; HSP w/o SR: HSP without semantic representation, HSP degrades to a two stage model; HSP w/o DR: Remove decomposed representation from HSP, it degrades to a two stage model; SP Unit: Remove all intermediate representations, HSP degrades to SP Unit.

First, we compare the HSP model with HSP(Switch), we observe that HSP outperforms HSP(Switch) by 0.89%, suggesting that parsing the intermediate representations in a bottom up way is more effective. Then we analyze the effect of different intermediate representations by removing semantic representation or sub-questions from HSP, resulting in performance degradation of 1.7% and 3.02% separately. Results prove that the question decomposition stage is most critical in HSP process. Finally, without any intermediate representations(the model degrades to a SP unit model), performance drops by 6.27% compared to our full HSP model.

4.4 Discussion and Analysis

4.4.1 Performance on Different Question Types

To evaluate the impact of question types on model performances, we calculate logical form accuracy on each type of questions of test set. Results are shown in Figure 3. Note that the test set consists of roughly 45% composition questions, 45% conjunction questions, 5% comparative questions and 5% superlative questions.

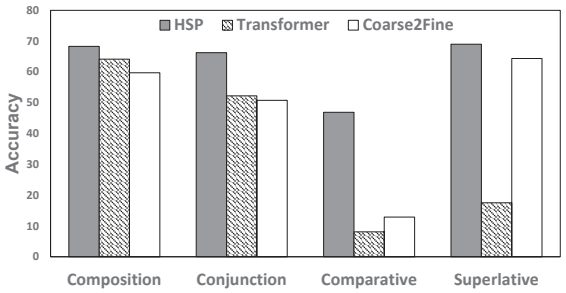

Figure 3: Model performances on different question types.

As Figure 3 shows, HSP has highest accuracy on the four type of question samples among the three models. Moreover, the accuracy of Transformer on composition and conjunction questions is comparable to that of Coarse2Fine and lower than HSP, showing that the HSP mechanism helps improve modeling capability. Finally, compared to Transformer, the accuracy of Coarse2Fine and HSP in comparative and superlative questions has been significantly improved, because these two models utilize additional information to enhance the robustness of the model, thus obtaining better results on types with much fewer training samples.

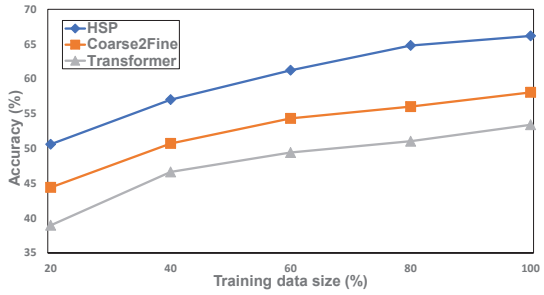

Figure 4: Model performances with different amount of training data.

Model	Dev	Test
PointerNetwork	31.3 / 55.2	31.9 / 55.7
PointerGenerator	55.5 / 69.3	55.0 / 69.0
HSP(SR)	**81.2 / 90.6**	**78.9 / 88.7**
w/o SR	78.9 / 88.5	76.3 / 86.8
w/o POS	78.3 / 88.1	75.8 / 86.3
w/o Glove	77.2 / 87.4	75.4 / 85.6
w/o Copy	73.7 / 86.5	71.3 / 84.6

Table 3: Bleu-4/Rouge-L scores on ComplexWebQuestions dataset, question decomposition task.

4.4.2 Performance on Different Training Data Volumes

In Figure 4, we depict the trends of test set accuracy with different portions of training data. The results of this experiment demonstrate that the performance of the HSP exceeds the other two baselines, regardless of the amount of training data. Moreover, as training data volume increases, the performance improvement that HSP can achieve is higher than the other two models. We think the reason is that as the training resources increase, HSP learns better question decomposer and information extractor and generates more accurate sub-questions and key information, which help HSP semantic parser to obtain better logical form results.

4.5 Question Decomposition Results

To further evaluate the effectiveness and generalization ability of our HSP model, we conduct question decomposition experiment with an HSP model variant and compare its performance to several neural models. We use case-insensitive Bleu-4 (Papineni et al., 2002) and Rouge-L (Lin, 2004) as evaluation metrics for question decomposition. For all models, the input is the complex question, and the output is decomposed sub-question sequence with the same format as decomposed representation.

Table 3 shows the question decomposition results of different models. PointerNetwork refers to the model (Talmor and Berant, 2018) on splitting the complex question into sub-questions using splitting points predicted by a pointer network model (Vinyals et al., 2015). HSP(SR) refers to a two-stage HSP model for which we use semantic representation as intermediate representation. We observe that compared to PointerNetwork, the other two models obtain much better results, proving that compared to decomposing complex question by finding splitting points in the question, our neural generative question decomposer is more effective. The Pointer-Generator follows the settings in semantic parsing experiments, and it obtains 55.0 Bleu-4 and 69.0 Rouge-L scores, which are lower than HSP. With the help of semantic representation and HSP model, HSP(SR) achieves 81.2/90.6 Bleu-4/Rouge-L scores on the test set, much higher than the two baselines.

We also perform ablation experiments on question decomposition to measure the impact of different modules, the results are also shown in Table 3. We examine four main modules in the HSP model: semantic representation(SR), POS embedding(POS), pre-trained Glove word embedding(Glove) and copy mechanism(Copy), and incrementally remove these modules from HSP(SR). Results show that without semantic representation in HSP, the model's Bleu-4 score decreases 2.6 points and the Rouge-L score decreases 1.9 points. The decrease of Bleu-4 score by removing HSP is only lower than removing the copy mechanism(4.1 points), and Rouge degradation is highest among the four ablation models. It indicates that HSP mechanism is vital for the model.

5 Conclusion

In this work, we propose a novel hierarchical semantic parsing (HSP) model based on sequence-to-sequence paradigm. Experiments show that compared to several previous systems, HSP effectively improves performance. We also design a neural generative question decomposer which achieves much higher performance than splitting-based question decomposition approach. Further experiments also prove that the proposed neural generative question decomposer also benefits from the HSP mechanism.

Acknowledgments

This research was supported by the National Key Research and Development Program of China (2018YFB1004502) and the National Natural Science Foundation of China (61690203, 61532001). We thank the anonymous reviewers for their helpful comments.

References

Yoav Artzi, Nicholas FitzGerald, and Luke S. Zettlemoyer. 2013. Semantic parsing with combinatory categorial grammars. In *ACL 2013*, page 2.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2014. Semantic parsing on freebase from question-answer pairs. In *EMNLP 2014*.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL 2013*, pages 423–433.

Bo Chen, Le Sun, and Xianpei Han. 2018. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In *ACL 2018*, pages 766–777. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *ACL 2016*, pages 33–43.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *ACL 2018*, pages 731–742.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In *ACL 2018*, pages 743–753.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. *arXiv preprint arXiv:1706.04326*.

Matt Gardner, Pradeep Dasigi, Srinivasan Iyer, Alane Suhr, and Luke Zettlemoyer. 2018. Neural semantic parsing. In *ACL 2018*, pages 17–18.

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. 2018. Weakly supervised semantic parsing with abstract examples. In *ACL 2018*, pages 1809–1819.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *ACL 2016*.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *ACL 2017*, pages 623–628. Association for Computational Linguistics.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2016. Answering complicated question intents expressed in decomposed question sequences. *arXiv preprint arXiv:1611.01242*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *ACL 2016*, pages 12–22. Association for Computational Linguistics.

Aditya Kalyanpur, Siddharth Patwardhan, BK Boguraev, Adam Lally, and Jennifer Chu-Carroll. 2012. Fact-based question decomposition in deepqa. *IBM Journal of Research and Development*, 56(3.4):13–1.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP 2013*, pages 1545–1556.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *EMNLP 2011*, pages 1512–1523.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *ACL 2011*, pages 590–599.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL 2014*, pages 55–60.

Raymond J. Mooney. 2007. Learning for semantic parsing. In *CICLing 2007*, pages 311–324.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL 2002*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP 2014*, pages 1532–1543.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the ACL*, 2(1):377–392.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL 2017*, pages 1073–1083. Association for Computational Linguistics.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR 2016*, pages 2818–2826.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS 2017*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS 2015*, pages 2692–2700.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *ACL 2016*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *CoRR*, abs/1711.04436.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *NAACL-HLT 2015*, pages 66–70.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL 2015*, pages 1321–1331. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI 2005*, pages 658–666.