Reranking for Neural Semantic Parsing

Pengcheng Yin
Language Technologies Institute
Carnegie Mellon University
pcyin@cs.cmu.edu

Graham Neubig
Language Technologies Institute
Carnegie Mellon University
gneubig@cs.cmu.edu

Abstract

Semantic parsing considers the task of transducing natural language (NL) utterances into machine executable meaning representations (MRs). While neural network-based semantic parsers have achieved impressive improvements over previous methods, results are still far from perfect, and cursory manual inspection can easily identify obvious problems such as lack of adequacy or coherence of the generated MRs. This paper presents a simple approach to quickly iterate and improve the performance of an existing neural semantic parser by reranking an n-best list of predicted MRs, using features that are designed to fix observed problems with baseline models. We implement our reranker in a competitive neural semantic parser and test on four semantic parsing (GEO, ATIS) and Python code generation (DJANGO, CONALA) tasks, improving the strong baseline parser by up to 5.7% absolute in BLEU (CONALA) and 2.9% in accuracy (DJANGO), outperforming the best published neural parser results on all four datasets.

1 Introduction

Semantic parsing is the task of mapping a natural language utterance into machine executable meaning representations (*e.g.*, Python code). Recent years have witnessed a burgeoning of applying neural network architectures for semantic parsing, from sequence-to-sequence models (Jia and Liang, 2016; Ling et al., 2016; Liang et al., 2017; Suhr et al., 2018), to more complex parsing paradigms guided by the structured topologies of target meaning representations (Xiao et al. (2016); Dong and Lapata (2016); Yin and Neubig (2017); Rabinovich et al. (2017); Krishnamurthy et al. (2017); Zhong et al. (2017); Dong and Lapata (2018); Iyer et al. (2018), *inter alia*).

While neural network-based semantic parsers have achieved impressive results, there is still room for improvement. A pilot analysis of incorrect predictions from a competitive neural semantic parser, TRANX (Yin and Neubig, 2018) indicates an obvious issue of *incoherence*. In the real example in Figure 1, top prediction $\boldsymbol{z}_1$ is semantically incoherent with the intent expressed in the utterance. Perhaps a more interesting issue is *inadequacy* — while the predicted MRs match the overall intent of the utterance, they still miss or misinterpret crucial pieces of information (*e.g.*, missing or generating wrong arguments, as in $\boldsymbol{z}_2$ and $\boldsymbol{z}_9$). Indeed, we observe that around 41% of the failure cases of TRANX on a popular Python code generation task (DJANGO, Oda et al. (2015)) are due to such inadequate predictions.

Although the top predictions from a semantic parser could fall short in adequacy or coherence, we found the parser still maintains high recall, covering the gold-standard MR in its n-best list of predictions most of the time[1]. This naturally motivates us to investigate whether the performance

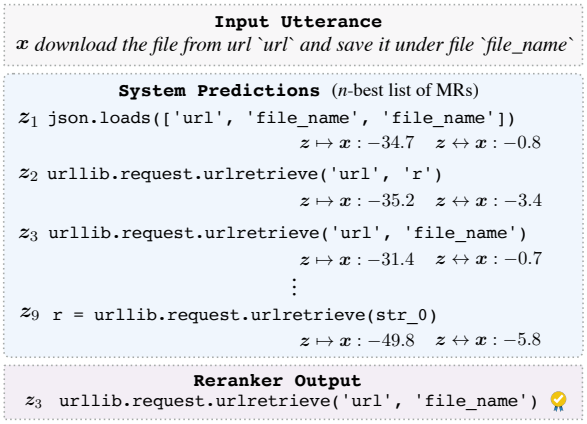

Figure 1: Illustration of the reranker with a real example from the CONALA code generation task (Yin et al., 2018a) with reconstruction ($\boldsymbol{z} \mapsto \boldsymbol{x}$) and discriminative matching ($\boldsymbol{x} \leftrightarrow \boldsymbol{z}$) scores.

[1] As we will show in § 3, our base neural semantic parser registers 77.3% top-1 accuracy and 84.0% recall over the 15-best beam on the DJANGO dataset, a **6.7%** absolute gap.

of an existing neural parser can be potentially improved by reranking the n-best list of candidate MRs. In this paper, we propose a simple reranker powered mainly by two quality-measuring features of a candidate MR: (1) a generative *reconstruction model*, which tests the coherence and adequacy of an MR via the likelihood of reconstructing the original input utterance from the MR; and (2) a discriminative *matching model*, which directly captures the semantic coherence between utterances and MRs. We implement our reranker in a strong neural semantic parser and evaluate on both tasks of parsing NL to domain-specific logical form (GEO, ATIS) and general-purpose source code (DJANGO, CONALA). Our reranking approach improves upon this strong parser by up to 5.7% absolute in BLEU (CONALA) and 2.9% in accuracy (DJANGO), outperforming the best published neural parser results on all datasets.

2 Reranking Model

Figure 1 illustrates our approach. Given an input NL utterance $\boldsymbol{x}$, we assume access to an existing neural semantic parser $p(\boldsymbol{z}|\boldsymbol{x})$, which outputs a ranked n-best list of system-generated meaning representations given $\boldsymbol{x}$, $\{\boldsymbol{z}_i\}_{i=1}^n$. In practice, such an n-best list is usually generated by approximate inference like beam search. The reranker $R(\cdot)$ takes as input the n-best list of MRs and the input utterance, and outputs the MR $\hat{\boldsymbol{z}}$ with the highest reranking score, *i.e.*, $\hat{\boldsymbol{z}} = \text{argmax}_{\boldsymbol{z}\in\{\boldsymbol{z}_i\}_{i=1}^n} R(\boldsymbol{z}, \boldsymbol{x})$. We parameterize $R(\cdot)$ as a (log-) linear model:

$$R(\boldsymbol{z}, \boldsymbol{x}) = \alpha_0 \log p(\boldsymbol{z}|\boldsymbol{x}) + \sum_{k=1}^{K} \alpha_k f_k(\boldsymbol{z}, \boldsymbol{x}), \quad (1)$$

where f_k is a feature function that scores a candidate prediction $\boldsymbol{z}$, and $\{\alpha\}$ tuned weights. We also include the original parser score in $R(\cdot)$. The idea of reranking the beam of candidate parses has been attempted for various NLP tasks (Collins and Koo, 2000), and was also previously applied for classical grammar-driven semantic parsers. Such reranking models typically use domain-specific syntactic features strongly coupled with the underlying parsing algorithm (*e.g.*, an indicator feature for each grammar rule applied, Raymond and Mooney (2006); Srivastava et al. (2017)), while our reranker applies domain-general quality-measuring features compatible with both domain-specific (*e.g.*, λ-calculus) and general-purpose (*e.g.*, Python) MRs (more in § 3). Specifically, our reranker mainly uses two features, whose scores are given by two external models: a *reconstruction model* and a *matching model*.

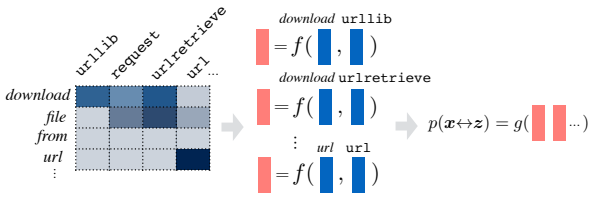

Figure 2: Illustration of the discriminative matching model, adapted from Parikh et al. (2016). Punctuations (dots, parentheses) in MR are omitted for clarity.

Generative Reconstruction Feature Our reconstruction feature $\log p(\boldsymbol{z} \mapsto \boldsymbol{x})$ is a generative model that scores the coherence and adequacy of an MR $\boldsymbol{z}$ using the probability of reproducing the original input utterance $\boldsymbol{x}$ from $\boldsymbol{z}$. Intuitively, a good candidate MR should adequately encode the semantics of $\boldsymbol{x}$, leading to high reconstruction score. The idea of using *reconstruction* as a quality metric is closely related to reconstruction models in auto-encoders (Vincent et al., 2008), and its applications in semi-supervised (Yin et al., 2018b) and weakly supervised (Cheng and Lapata, 2018) semantic parsing, where $p(\boldsymbol{z}\mapsto\boldsymbol{x})$ is used to score the quality of sampled MRs in optimization. Similar models have also been applied for pragmatic inference in instruction-following agents for modeling the likelihood of causing the speaker to produce the utterance given an inferred action (Fried et al., 2018), while we use $p(\boldsymbol{z}\mapsto\boldsymbol{x})$ as one quality-measuring feature in our reranker.

Specifically, we implement $p(\boldsymbol{z} \mapsto \boldsymbol{x})$ using an attentional sequence-to-sequence network (Luong et al., 2015), which takes as input a tokenized MR $\boldsymbol{z}$. The network is augmented with a copy mechanism (Gu et al., 2016), allowing out-of-vocabulary variable names (*e.g.*, `file_name` in Figure 1) in $\boldsymbol{z}$ to be directly copied to the utterance $\boldsymbol{x}$.

Discriminative Matching Feature We use a matching model to measure the probability of the input utterance $\boldsymbol{x}$ and a candidate MR $\boldsymbol{z}$ being semantically coherent to each other. Intuitively, for a semantically coherent parse $\boldsymbol{z}$ (*e.g.*, $\boldsymbol{z}_3$ in Figure 1), each sub-piece in $\boldsymbol{z}$ (*e.g.*, `urllib.request.urlretrieve`) could coarsely match with a span (*e.g.*, *download the file*) in the utterance, and *vice versa*. Motivated by this observation, we implement $p(\boldsymbol{x}\leftrightarrow\boldsymbol{z})$ as a decompos-

able attention model (Parikh et al., 2016), a discriminative model which computes a semantic coherence score based on the latent pairwise alignment between tokens in $\boldsymbol{x}$ and $\boldsymbol{z}$. Figure 2 depicts an overview of the model, while we refer interested readers to Parikh et al. (2016) for technical details. Intuitively, the model measures the semantic equivalence of an utterance $\boldsymbol{x}$ and an MR $\boldsymbol{z}$ based on pair-wise associations of tokens in $\boldsymbol{x}$ and $\boldsymbol{z}$ in three steps: (1) an alignment step, where alignment scores between each pair of tokens in $\boldsymbol{x}$ and $\boldsymbol{z}$ are computed using attention; (2) a comparison step, where a set of representations are produced from embeddings of aligned tokens, capturing their semantic similarities; and (3) an aggregation step, where all pairwise comparisons are combined to compute the semantic coherence score.

Token Count Feature Besides the two primary features introduced above, we also include an auxiliary *token count* feature $|\boldsymbol{z}|$ of an MR, which has been shown useful in preventing a machine translation model from favoring shorter predictions (Cho et al., 2014; Och and Ney, 2002), while we test them for reranking MRs, especially when the target metric is BLEU (§ 3).

3 Experiment

We test on four semantic parsing and code generation benchmarks:

GEO (Zelle and Mooney, 1996) and **ATIS** (Deborah A. Dahl and Shriber) are two closed-domain semantic parsing datasets. The NL utterances are geographical (GEO) and flight booking (ATIS) inquiries (*e.g.*, *What is the latest flight to Boston?*). The corresponding MRs are defined in λ-calculus logical forms (*e.g.*, `argmax` x `(and (flight` x`) (to` x `boston)) (departure_time` x`))`).

DJANGO (Oda et al., 2015) is a popular Python code generation dataset consisting of NL-annotated code from the Django framework. Around 70% of examples are simple cases of variable assignment (*e.g.*, `result = []`), function definition/invocation or condition tests, which can be easily inferred from the verbose NL utterances (*e.g.*, *Result is an empty list*).

CONALA (Yin et al., 2018a)[2] is a newly introduced task for open-domain code generation. It consists of 2,879 examples of manually annotated NL questions (*e.g.*, *Check if all elements in list 'my_list' are the same*) and their Python solution (*e.g.*, `len(set(mylist)) == 1`) on STACK OVERFLOW. Compared with DJANGO, examples in CONALA cover real-world NL queries issued by programmers with diverse intents, and therefore are significantly more difficult due to its broad coverage and high compositionality of target MRs.

Base Semantic Parser $p(\boldsymbol{z}|\boldsymbol{x})$ While we remark that our reranking model is parser agnostic, in the experiments we are primarily interested in investigating if the reranker could further improve the performance of an already-strong semantic parser. We use TRANX (Yin and Neubig, 2018)[4], a general-purpose open-source neural semantic parser that maps an input utterance into MRs using a neural sequence-to-tree network, where MRs are represented as abstract syntax trees. We leave evaluating the performance of the reranker with other parsers as interesting future work.

Training Reranking Model Deploying the reranker to a benchmark dataset involves three steps: (1) training the base parser, (2) training the reranking features (reconstruction and matching models), and (3) tuning the feature weights.

(1) Training Base Parser We use its pre-processed version of the dataset shipped with the base parser TRANX. We train TRANX using its official configuration and collect the n-best list of candidates for each example using beam search (beam size is 5 for GEO and ATIS, and 15 for DJANGO and CONALA).

(2) Training Reranking Features The reconstruction model is trained using standard maximum-likelihood estimation using utterances and their associated gold MRs in the training set. We then chose the model with the lowest perplexity on the development set[5]. To train the matching model, it requires training examples in the form of triplets $\langle \boldsymbol{x}, \boldsymbol{z}, y \rangle$, consisting of an utterance $\boldsymbol{x}$, an MR $\boldsymbol{z}$ and a binary label y indicating whether $\boldsymbol{z}$ is a correct parse of $\boldsymbol{x}$. During optimization we create a mini-batch $\mathcal{D}$ by: (a) sampling a mini-batch of 10 gold examples $\langle \boldsymbol{x}, \boldsymbol{z}^* \rangle$ ($\boldsymbol{x}$ and its correct MR $\boldsymbol{z}^*$) from the original training set and adding those examples (with $y = 1$) to $\mathcal{D}$; (b) for each

[2] https://conala-corpus.github.io/

[4] http://pcyin.me/tranX

[5] GEO does not have a development set, so we randomly sample 20% examples from its official training set for development, and use the remaining 80% examples for training the reranker and its features.

GEO		ATIS		DJANGO		CONALA	
Model	ACC.	Model	ACC.	Model	ACC.	Model	BLEU / ACC.
RSK17	87.1	RSK17	85.9	YN17	71.6	SEQ2SEQ†	10.58 / n/a
DL18	88.2	DL18	87.7	DL18	74.1	Best Submission†	24.30 / n/a
Base Parser	88.8 ±1.0	Base Parser	87.6 ±0.1	Base Parser	77.3 ±0.4	Base Parser	24.35 ±0.4 / 2.5 ±0.7
+ Reranker	89.1 ±1.2	+ Reranker	**88.4** ±0.5	+ Reranker	**80.2** ±0.4	+ Reranker	**30.11** ±0.6 / 2.8 ±0.5
– *recon.*	88.9 ±0.9	– *recon.*	87.8 ±0.3	– *recon.*	78.1 ±0.3	– *recon.*	28.41 ±1.0 / 2.1 ±0.5
– *match.*	89.2 ±1.0	– *match.*	87.7 ±0.5	– *match.*	79.9 ±0.4	– *match.*	29.89 ±0.5 / 2.6 ±0.6
– *t.c.*	**89.4** ±0.9	– *t.c.*	88.1 ±0.7	– *t.c.*	80.0 ±0.4	– *t.c.*	28.45 ±1.1 / **3.0** ±0.5
Oracle	90.9 ±0.6	Oracle	90.4 ±0.3	Oracle	84.0 ±0.6	Oracle	37.08* ±0.6 / 5.4 ±0.6

Table 1: Mean and standard deviation[3] over five random runs. *recon.*, *match.*, *t.c.* stand for reconstruction, matching, and token count features, resp. †Results taken from the official CONALA leaderboard as on Feb. 25th, 2019. *Upper-bound corpus-BLEU is approximated by choosing $\boldsymbol{z}$ in the n-best list with the highest sentence-level BLEU.

$\boldsymbol{x}$ in the batch, sampling an incorrect MR $\boldsymbol{z}'$ from the n-best list, and adding the negative example $\langle \boldsymbol{x}, \boldsymbol{z}', y = 0 \rangle$ to $\mathcal{D}$. Validation is performed by evaluating the classification accuracy on development set constructed similarly.

We also made one special modification for the discriminative matching model on DJANGO. Different with the preprocessed version of other datasets, OOV variable names in DJANGO cannot be easily identified and canonicalized, which hurts the performance of the vanilla matching model. Therefore, for each input $\langle \boldsymbol{x}, \boldsymbol{z} \rangle$, we replace each OOV token (*e.g.*, a variable name `my_list`) in $\boldsymbol{x}$ and $\boldsymbol{z}$ with a unique numbered slot (*e.g.*, `VAR0`). Hence, different OOV variable names in the input can still be distinguished based on their slot IDs. We found this simple trick improved the average classification accuracy on the development set from 77% to 81%.

(3) Tuning Feature Weights Finally, given the trained features, we then tune the feature weights in E.q. 1 using the minimum risk training (Smith and Eisner, 2006) algorithm implemented in the `Travatar` package (Neubig, 2013), which optimizes the expected metric over candidates in the n-best list of candidate MRs on development sets. Steps (2) and (3) are quite efficient, and takes less than 10 minutes on a server with a GPU.

Metric We use the standard evaluation metric for each dataset: exact match **accuracy** for GEO, ATIS, DJANGO and corpus-level **BLEU** for CONALA.

3.1 Results

Table 1 lists evaluation results. We also report the oracle recall over n-best list as an upper-bound performance (last line in Table 1). First, we note that our base parser is indeed strong, performing competitively against existing neural systems on all datasets. This suggests that our base parser will serve as a reasonable testbed for the reranking model. Next, we observe that the reranker achieves improved results across the board, closing the gap between top-1 predictions and oracle recall. Notably, the reranker registers 2.9% absolute gain in accuracy on DJANGO and 5.7% in BLEU on CONALA, resp. This demonstrates that reranking is an effective approach to improve the performance of an already-strong neural parser.

We also performed a feature ablation study by removing one feature at a time. For discussion, we also present qualitative examples of reranking results in Table 2. We are particularly interested in investigating the comparative utility of the discriminative matching and reconstruction features. Interestingly, we observe that while the matching feature seems to be important for semantic parsing tasks like ATIS, the reconstruction model performs generally better on two Python code generation tasks, where target MRs are much more complex. We hypothesize that our matching model based on pair-wise token associations between utterances and MRs is particularly effective for simpler MRs in ATIS, where there is a clear correspondence between utterance spans (*e.g.*, *round trip* in Example 1, Table 2) and MR predicates (*e.g.*, `round_trip`). This could also hold for some examples in DJANGO, where the verbose NL utterances could be roughly aligned with the MR (*e.g.*, Example 2). On the other hand, we observe the reconstruction model could potentially go beyond surface token-wise match between NL utterances and MRs, promoting more complex (longer)

[5] We observe relatively high variance on GEO, possibly due to its small size (599/279 train/test examples).

ATIS *Show me all round trip flight from ci0 to ci1 nonstop*

z_1 `lambda $0 e (and (flight $0) (from $0 ci0) (to $0 ci1) (nonstop $0))` ✗

$p(z|x) = -2.0 \quad z \mapsto x = -11.1 \quad x \leftrightarrow z = -14.3$

z_2 `lambda $0 e (and (flight $0) (from $0 ci0) (to $0 ci1) (nonstop $0) (round_trip $0))` ✓

$p(z|x) = -2.3 \quad z \mapsto x = -4.2 \quad x \leftrightarrow z > -0.01$

DJANGO *If length of version does not equals to integer 5, raise an exception*

z_1 `raise` ✗

$p(z|x) = -0.1 \quad z \mapsto x = -61.7 \quad x \leftrightarrow z = -5.3$

z_2 `assert len(version) == 5` ✓

$p(z|x) = -3.3 \quad z \mapsto x = -17.0 \quad x \leftrightarrow z = -0.5$

DJANGO *and truncate, return the result. return elements of words joined in a string, separated with whitespace*

z_1 `return str('joined')` ✗

$p(z|x) = -1.6 \quad z \mapsto x = -54.6 \quad x \leftrightarrow z = -0.68$

z_5 `return ' '.join(words)` ✓

$p(z|x) = -2.6 \quad z \mapsto x = -48.8 \quad x \leftrightarrow z = -0.61$

CONALA *Removing duplicates in list 't'*

z_1 `print(list(itertools.chain(*t)))` ✗

$p(z|x) = -5.9 \quad z \mapsto x = -15.5 \quad x \leftrightarrow z = -0.4$

z_4 `list(set(t))` ✓

$p(z|x) = -6.7 \quad z \mapsto x = -11.3 \quad x \leftrightarrow z = -1.4$

Table 2: Examples with original parser score $p(z|x)$, reconstruction ($z \mapsto x$) and discriminative matching ($x \leftrightarrow z$) feature scores (*log* scale). We show both the original top prediction z_1 and the highest-scored one z_i by the reranker.

candidate MRs that more adequately encode the semantics of the input utterance[6] (*e.g.*, in Example 3, z_5 receives a much higher reconstruction score, while the difference between the discriminative matching scores is small). Therefore, on the challenging CONALA dataset with much weaker alignment between its succinct NL utterances and highly compositional MRs (*e.g.*, Example 4), the matching model does not function as well (*e.g.*, the incorrect MR z_1 received a higher matching score). While more careful investigation of the relative advantage between the reconstruction and discriminative matching features remain an interesting future work, we remark that the reconstruction model $p(z \mapsto x)$, when combined with with the original parser score $p(z|x)$ in E.q. (1), also implicitly functions as a matching model that measures the semantic similarity using the bidirectional generation likelihood between x and z. Such architecture could be an interesting future direction for modeling semantic similarity.

Additionally, the auxiliary token count feature is also effective, especially on CONALA, yielding a +1.66 gain in BLEU by promoting longer MRs.

Finally, we investigated the failure cases where our best-performed reranker generated incorrect MRs. We are particularly interested in those remaining failed examples on simpler semantic parsing tasks (GEO, ATIS), where our reranker's accuracies are close to the oracle recall. For instance, on ATIS, only 6 incorrect examples (out of a total of 49) are due to reranking error, 10 are due to that the gold MRs are not included the n-best list, while most (20) remaining cases are because the specific NL patterns in testing utterances (*e.g.*, the temporal NL pattern *flight ... prior to ...*) are not covered by its (small) training set. This interesting result suggests that incorporating external linguistic knowledge (*e.g.*, Wang et al. (2014)) is important in order to further improve the performance of neural parsers on closed-domain semantic parsing tasks.

[6] On DJANGO, the average length of top-reranked MRs by the reconstruction and matching models is 8.6 and 7.7, resp.

4 Conclusion

We proposed a feature-based reranker for neural semantic parsing, which achieved strong results on three semantic parsing and code generation tasks. In the future we plan to apply the reranker to other parsers and more benchmark datasets. We will also attempt to jointly train the base semantic parser and the reranker by using the reranker's output as supervision to fine tune the base parser.

Acknowledgments

This research was supported by NSF Award No. 1815287 "Open-domain, Data-driven Code Synthesis from Natural Language," and a grant from the Carnegie Bosch Institute.

References

Jianpeng Cheng and Mirella Lapata. 2018. Weakly-supervised neural semantic parsing with a generative ranker. In *CoNLL*.

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

Michael Collins and Terry K Koo. 2000. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31:25–70.

Michael Brown William Fisher Kate Hunicke-Smith David Pallett Christine Pao Alexander Rudnicky Deborah A. Dahl, Madeleine Bates and Elizabeth Shriber.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*.

Li Dong and Mirella Lapata. 2018. coarse-to-fine decoding for neural semantic parsing. In *Proceedings of ACL*.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018. Unified pragmatic models for generating and following instructions. In *Proceedings of NAACL*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2018. Mapping language to code in programmatic context. In *Proceedings of EMNLP*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of EMNLP*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of ACL*.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of ACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Graham Neubig. 2013. Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proceedings of the ACL Demonstration Track*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation (T). In *Proceedings of ASE*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of EMNLP*.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of ACL*.

Ruifang Ge Raymond and J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of COLING/ACL*.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL*.

Shashank Srivastava, Amos Azaria, and Tom Michael Mitchell. 2017. Parsing natural language conversations using contextual cues. In *IJCAI*.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of NAACL-HLT*.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*.

Adrienne Wang, Tom Kwiatkowski, and Luke Zettlemoyer. 2014. Morpho-syntactic lexical generalization for CCG semantic parsing. In *Proceedings of EMNLP*.

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

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018a. Learning to mine aligned code and natural language pairs from stack overflow. In *Proceedings of MSR*.

Pengcheng Yin and Graham Neubig. 2017. a syntactic neural model for general-purpose code generation. In *Proceedings of ACL*.

Pengcheng Yin and Graham Neubig. 2018. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of EMNLP Demonstration Track*.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018b. StructVAE: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of ACL*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, pages 1050–1055.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Reranking for Neural Semantic Parsing
Supplementary Materials

A Reranker Model Configuration and Training

For each benchmark dataset, we use its pre-processed version of the dataset shipped in the base parser. We first train the base semantic parser using its official configuration for the dataset. We then generate the n-best list of candidates for reranking using beam search (beam size is 5 for GEO and ATIS, and 15 for DJANGO and CONALA). For the reranking model, we first train the reconstruction and paraphrase identification models on the training set, and then tune the feature weights on the development set.[7]

Reconstruction Model We train the construction model $p(\boldsymbol{z} \mapsto \boldsymbol{x})$ on each dataset using the training examples, and choose the model with the lowest perplexity on the development set. The dimensionality of embedding layers and RNN hidden layers is 128 and 256, respectively.

Paraphrase Identification Model We implement the decomposable attention model following the reference implementation in the `AllenNLP` toolkit, with a word embedding size of 128. We also made one special modification for DJANGO. Different from the preprocessed version of other datasets shipped with the base parser, OOV variable names in DJANGO cannot be easily identified and canonicalized, which affects the performance of the vanilla paraphrase identification model. Therefore, for each input $\langle \boldsymbol{x}, \boldsymbol{z} \rangle$ to $p(\boldsymbol{x} \leftrightarrow \boldsymbol{z})$, we replace each OOV token (*e.g.*, a variable name `my_list`) in $\boldsymbol{x}$ and $\boldsymbol{z}$ with a unique numbered slot (*e.g.*, `VAR0`). Hence, different OOV variable names in the input can still be distinguished based on their slot IDs. We found this simple trick improved the average paraphrase identification accuracy on the development set from 77% to 81%.

The paraphrase model requires training instances of $\langle \boldsymbol{x}, \boldsymbol{z}, y \rangle$ consisting of triplets of utterance $\boldsymbol{x}$, MR $\boldsymbol{z}$ and the corresponding binary label y indicating whether $\boldsymbol{x}$ and $\boldsymbol{z}$ is semantically equivalent. To create the training data, we took the n-best list of decoded MRs $\mathcal{H}(\boldsymbol{x})$ for each training instance $\boldsymbol{x}$. During training, we construct each batch $\mathcal{D}$ by first sampling 10 pairs of utterances $\boldsymbol{x}$ and its gold-standard MRs $\boldsymbol{z}$ from the training set, and adding the sampled $\langle \boldsymbol{x}, \boldsymbol{z}, y = 1 \rangle$ into $\mathcal{D}$. We then generate negative examples by taking the incorrect MR $\boldsymbol{z}' \in \mathcal{H}(\boldsymbol{x})$ with the highest parser score $p(\boldsymbol{z}|\boldsymbol{x})$ for each utterance $\boldsymbol{x}$ in the batch, and add the negative samples $\langle \boldsymbol{x}, \boldsymbol{z}', y = 0 \rangle$ into $\mathcal{D}$. We perform validation by evaluating paraphrase identification accuracy on the development set constructed similarly.

Tuning Feature Weights Once the reconstruction and paraphrase models are trained, we tune the feature weights $\{\alpha\}$ in E.q. (1) on the development set using minimum risk training (Smith and Eisner, 2006), which optimizes the expected metric (accuracy for GEO, ATIS and DJANGO, and corpus-BLEU for CONALA) over the n-best list of candidate MRs. We use the readily-available implementation in the reranking module of the `Travatar` package (Neubig, 2013).

[7] GEO does not have a development set, so we randomly sample 20% examples from its official training set for development, and use the remaining 80% examples for training the reranker and its features.