PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models

Torsten Scholak and **Nathan Schucher** and **Dzmitry Bahdanau**
ElementAI, a ServiceNow company
{torsten.scholak,dzmitry.bahdanau}@servicenow.com

Abstract

Large pre-trained language models for textual data have an unconstrained output space; at each decoding step, they can produce any of 10,000s of sub-word tokens. When fine-tuned to target constrained formal languages like SQL, these models often generate invalid code, rendering it unusable. We propose PICARD[1], a method for constraining auto-regressive decoders of language models through incremental parsing. PICARD helps to find valid output sequences by rejecting inadmissible tokens at each decoding step. On the challenging Spider and CoSQL text-to-SQL translation tasks, we show that PICARD transforms fine-tuned T5 models with passable performance into state-of-the-art solutions.

1 Introduction

While there have been many successes in applying large pre-trained language models to downstream tasks, our ability to control and constrain the output of these models is still very limited. Many enterprise applications are out of reach because they require a degree of rigour and exactitude that language models are not able to deliver yet. If the target is a formal language like SQL, then we would like the model to adhere exactly and provably to the SQL specification with all its lexical, grammatical, logical, and semantical constraints. Unfortunately, with pre-training alone, language models may not satisfy these correctness requirements.

For text-to-SQL translation, the most widespread solution to constrained decoding is to make invalid SQL unrepresentable. For a while now it has been possible to restrict auto-regressive decoding to only those token sequences that correctly parse to SQL abstract syntax trees (Yin and Neubig, 2018; Lin et al., 2019; Wang et al., 2020). More recently, semi-auto-regressive improvements to this parsing paradigm have been proposed (Rubin and Berant, 2021). However, while effective, these approaches have in common that they are achieved at the expense of using a custom vocabulary of special control tokens or a custom model architecture, or both. Unfortunately, this makes them incompatible with generic pre-trained language model decoders. A less invasive and more compatible approach is to not constrain the generation process, but instead to filter finalized beam hypotheses by validity (Suhr et al., 2020; Lin et al., 2020). Yet, such filtering is at the expense of a very large beam size.

We address the expenses of these approaches with a novel incremental parsing method for constrained decoding called PICARD, which stands for "Parsing Incrementally for Constrained Auto-Regressive Decoding." PICARD is compatible with any existing auto-regressive language model de-

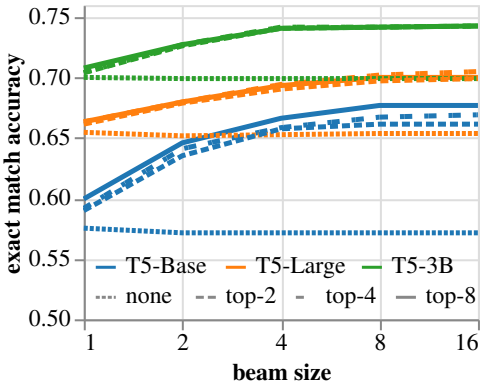

Figure 1: Exact-set-match accuracy of the highest-scoring prediction as a function of beam size on the Spider text-to-SQL development set. With PICARD turned on, token predictions had to pass PICARD checking at every decoding step. Only the top-2, -4, and -8 token predictions of each hypothesis were considered in the beam search. With PICARD turned off (none), all token predictions were considered and none were checked. The models, T5-Base, -Large, and -3B, did not have access to any database content, only to the database schemas.

[1]The PICARD code is available at https://github.com/ElementAI/picard.

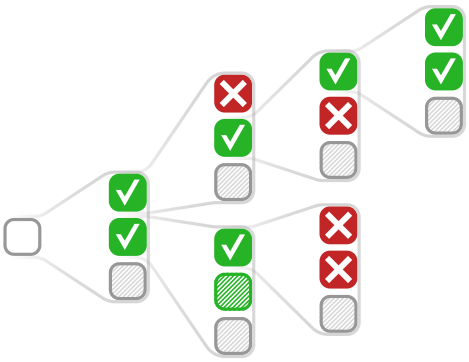

Figure 2: Illustration of constrained beam search with beam size 2 and PICARD. Each vertical column represents three token predictions for a hypothesis from top to bottom in descending order by probability. In this example, PICARD is configured to only check the top-2 highest ones. The rest is automatically dismissed by setting their score to $-\infty$. Tokens rejected by PICARD (red, $\times$) are also assigned a score of $-\infty$. Accepted tokens (green, ✓) keep their original score.

coder and vocabulary—including, but not limited to, those of large pre-trained transformers—and it does not require very large beam sizes. PICARD is entirely absent from pre-training or fine-tuning of the model, and can be easily and optionally enabled at inference time. PICARD operates directly on the output of the language model which, in the case of text-to-SQL translation, is the readable surface form of the SQL code.

In our experiments, we find that PICARD can significantly improve the performance of a large pre-trained language model (Raffel et al., 2020) after it is fine-tuned on the text-to-SQL task. On the Spider text-to-SQL dataset (Yu et al., 2018), we find that a T5-Base model with PICARD can outperform a T5-Large model without it, and likewise for a T5-Large and a T5-3B model. Significantly, with the help of PICARD, a T5-3B model can be raised to state-of-the-art performance on the Spider and CoSQL datasets (Yu et al., 2019).

2 The PICARD Method

PICARD warps model prediction scores and integrates trivially with existing algorithms for greedy and beam search used in auto-regressive decoding from language models. Its arguments are the token ids of the current hypothesis and, for each vocabulary token, the log-softmax scores predicted by the model's language modeling head. PICARD also has access to SQL schema information, in particular, information about the names of tables and columns and about which column resides in which table.

At each generation step, PICARD first restricts prediction to the top-k highest probability tokens and then assigns a score of $-\infty$ to those that fail PICARD's numerous checks (see Figure 2). These checks are enabled by fast incremental parsing (O'Sullivan and Gamari, 2021) based on monadic combinators (Leijen and Meijer, 2001). There are four PICARD mode settings that control their comprehensiveness: off (no checking), lexing, parsing without guards, and parsing with guards—the highest mode. A prediction that passes a higher mode will always pass a lower mode but not necessarily vice versa.

2.1 Lexing

In lexing mode, PICARD checks the output on a lexical level only. It attempts to convert the partial, detokenized model output to a white-space delimited sequence of individual SQL keywords like `select`, punctuation like `()`, operators like `+` and `-`, literals like string and number values in SQL conditions, and identifiers like aliases, tables, and columns—without being sensitive to the order in which these lexical items appear. By making it so, PICARD can detect spelling errors in keywords or reject table and column names that are invalid for the given SQL schema. For instance, consider the question "What are the email, cell phone and home phone of each professional?" from Spider's development set on the `dog_kennels` database. Our fine-tuned T5-Large model predicts `select email_address, cell_phone, home_phone from professionals` while the ground truth selects `cell_number` instead of the invalid `cell_phone` column. This mistake is caught and avoided by PICARD in lexing mode.

2.2 Parsing without Guards

In the lowest parsing mode above lexing—referred to as parsing without guards—PICARD checks the output on a grammatical level. PICARD attempts to parse the detokenized model output to a data structure that represents the abstract syntax tree (AST) of the predicted SQL query. Contrary to lexing mode, the order in which keywords and clauses appear now matters. PICARD can reject invalid query structures, e.g. find missing `from` clauses or incorrect orders of clauses and keywords. It can also detect a range of issues with compositions of SQL expressions: Number one, if PICARD matches on a `tid.cid` pattern, but the table with the id `tid`

does not contain a column with id `cid`, then that parse is rejected. Secondly, if PICARD first matches on an `alias.cid` pattern and then later matches on the `tid as alias` pattern but `tid` does not contain `cid`, then that parse is also rejected. An equivalent rule also exists for sub-queries bound to table aliases. Lastly, PICARD prohibits duplicate binding of a table alias in the same `select` scope, but permits shadowing of aliases defined in a surrounding scope. This can happen in nested SQL queries.

2.3 Parsing with Guards

In its highest parsing mode, PICARD engages in additional analyses—called guards—while assembling the SQL AST. If PICARD matches on `tid.cid` or `alias.cid`, then guards require that the table `tid` or the alias `alias`, respectively, is eventually brought into scope by adding it to the `from` clause. Moreover, the alias `alias` is constrained to resolve to a table or a sub-query that has the column `cid` in it. If PICARD matches on the pattern `cid`, then another guard requires that exactly one table is eventually brought into scope that contains a column with that id. These guards are enforced eagerly in order to fail fast and to eject invalid hypotheses from the beam at the earliest possible time. The first time this is happening is after parsing the `from` clause.

Only with these guards, PICARD is able to reject a wrong prediction from our fine-tuned T5-Large model like `select maker, model from car_makers` for the question "What are the makers and models?" Here, the correct table to use would have been `model_list`, since it is the only one in Spider's `car_1` schema that contains both a `maker` and a `model` column.

Additional checks and guards are conceivable, for instance, checking that only expressions of the same type are compared or that column types selected by `union`, `except`, or `intersect` queries match. We leave these additional checks to future work.

3 Experiments

Our experiments are mainly focused on Spider (Yu et al., 2018), a large multi-domain and cross-database dataset for text-to-SQL parsing. We train on the 7,000 examples in the Spider training set and evaluate on Spider's development set and its hidden test set. We also report results on the CoSQL SQL-grounded dialog state tracking task (Yu et al., 2019), where we predict a SQL query for each question given previous questions in an interaction context. For this task, we train on both the Spider text-to-SQL training data and the CoSQL dialog state tracking training data, and evaluate on the CoSQL development and test sets.

Spider and CoSQL are both zero-shot settings. There is no overlap between questions or databases between the respective training, development, and test sets.

On Spider, we determine model performance based on three metrics: exact-set-match accuracy, execution accuracy, and test-suite execution accuracy (Zhong et al., 2020). Exact-set-match accuracy compares the predicted and the ground-truth SQL query by parsing both into a normalized data structure. This comparison is not sensitive to literal query values and can decrease under semantic-preserving SQL query rewriting. Execution accuracy compares the results of executing the predicted and ground-truth SQL queries on the database contents shipped with the Spider dataset. This metric is sensitive to literal query values, but suffers from a high false positive rate (Zhong et al., 2020). Lastly, test-suite execution accuracy extends execution to multiple database instances per SQL schema. The contents of these instances are optimized to lower the number of false positives and to provide the best approximation of semantic accuracy.

On CoSQL, we measure model performance in terms of the question match accuracy and the interaction match accuracy. Both metrics are based on exact-set-match accuracy. Interaction match accuracy is the joint accuracy over all questions in an interaction.

We are encouraged by results by Shaw et al. (2021), who showed that a pre-trained T5-Base or T5-3B model can not only learn the text-to-SQL task, but also generalize to unseen databases, and even that T5-3B can be competitive with the then-state-of-the-art (Choi et al., 2021; Wang et al., 2020)—all without modifications to the model. We therefore use T5 as the baseline for all our experiments.

In order to allow for generalization to unseen databases, we encode the schema together with the questions. We use the same serialization scheme used by Shaw et al. (2021). In experiments using database content, we detect and attach the database values to the column names in a fashion similar to

We can assemble RA tree instead of SQL AST and incorporate this guards logic

Can brainstorm over additional checks to have

We can use T5, study and use FLAN-T5 as baseline

the BRIDGE model by Lin et al. (2020). When fine-tuning for the CoSQL dialog state tracking task, we append the previous questions in the interaction in reverse chronological order to the input. Inputs exceeding the 512-token limit of T5 are truncated. The target is the SQL from the Spider and/or CoSQL training sets, unmodified except for a conversion of keywords and identifiers to lower case. We fine-tune T5 for up to 3072 epochs using Adafactor (Shazeer and Stern, 2018), a batch size of 2048, and a learning rate of 10^{-4}.

Results Our findings on the Spider dataset are summarized in Table 1 and Figure 1. Our reproductions of Shaw et al. (2021)'s results with T5 cannot compete with the current state of the art on Spider. The issue is that these models predict a lot of invalid SQL. For instance, 12% of the SQL queries generated by the T5-3B model on Spider's development set result in an execution error. However, when these same models are augmented with PICARD, we find substantial improvements. First, invalid SQL predictions become rare. For T5-3B with PICARD, only 2% of the predictions are unusable. In these cases, beam search exited without finding a valid SQL prediction. Second, and most significantly, by using PICARD, the T5-3B model is lifted to state-of-the-art performance. We measure an exact-set-match accuracy of 75.5% on the development set and 71.9% on the test set. The execution accuracy results are 79.3% and 75.1%, respectively. These numbers are on par or higher than those of the closest competitor, LGESQL + ELECTRA (Cao et al., 2021) (see Table 1). Furthermore, we achieve a test-suite execution accuracy of 71.9% on Spider's development set.

Our findings on the CoSQL dialog state tracking dataset (see Table 2) are similar to those for Spider. PICARD significantly improves the performance, and our fine-tuned T5-3B model achieves state-of-the-art performance.

PICARD is not only improving performance, it is also fast. During evaluation of the T5-3B model on Spider, the decoding speed with beam size 4 on an NVIDIA A100-SXM4-40GB GPU was, on average, 2.5 seconds per sample without PICARD and 3.1 seconds per sample with PICARD.

Beam Size Figure 1 shows results on Spider without and with PICARD when parsing with guards for different beam sizes and sizes of T5. For each model size, PICARD increases performance

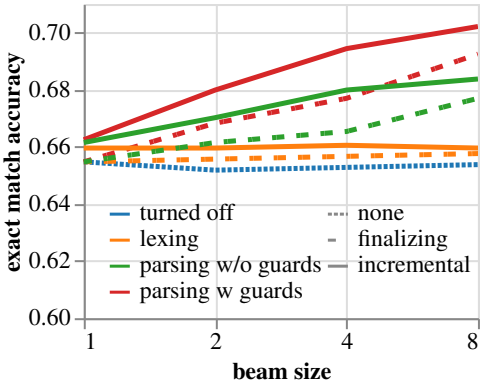

Figure 3: Exact-set-match accuracy on the Spider development set as a function of beam size for top-4 PICARD on T5-Large (schema only) and for different operation modes: turned off, lexing, parsing without guards, and parsing with guards. In each mode, PICARD is either used incrementally at each step or only when finalizing a hypothesis.

with increasing beam size. These increases are the strongest for the step from beam size 1 to 2, less pronounced from 2 to 4, and then saturating for beam sizes above 4. Even with greedy search (beam size 1), PICARD allows for some modest improvements. Note that, without PICARD, these models do not benefit from beam search. The number, k, of highest-probability tokens that are processed by PICARD at each decoding step has a modest to negligible impact on performance. It is the largest for T5-Base, smaller for T5-Large, and almost undetectable for T5-3B. We do not study the case $k = 1$, because it reduces the beam search to constrained greedy search.

Ablations In Figure 3, we have condensed our ablation analysis for PICARD. We show results for our T5-Large model in all four PICARD checking modes and for four different beam sizes on the Spider development set. When checking incrementally at each decoding step, lexing shows a small improvement over the unconstrained T5 model. The results without PICARD and with PICARD in lexing mode are largely independent of the beam size. This is different when PICARD is switched into the more sophisticated parsing modes. Both, with and without guards, improvements from PICARD increase rapidly for increasing beam sizes, where parsing with guards clearly has a strong lead over parsing without them.

In order to compare PICARD with the filtering-by-validity approach of Suhr et al. (2020) and Lin

System	Development		Test	
	EM%	EX%	EM%	EX%
BRIDGE v2 + BERT (ensemble)† (Lin et al., 2020)	71.1	70.3	67.5	68.3
SmBoP + GraPPa† (Rubin and Berant, 2021)	74.7	75.0	69.5	71.1
RATSQL + GAP† (Shi et al., 2021)	71.8	-	69.7	-
DT-Fixup SQL-SP + RoBERTa† (Xu et al., 2021)	75.0	-	70.9	-
LGESQL + ELECTRA† (Cao et al., 2021)	75.1	-	**72.0**	-
T5-Base (Shaw et al., 2021)	57.1	-	-	-
T5-3B (Shaw et al., 2021)	70.0	-	-	-
T5-Base (ours)	57.2	57.9	-	-
T5-Base+PICARD	65.8	68.4	-	-
T5-Large	65.3	67.2	-	-
T5-Large+PICARD	69.1	72.9	-	-
T5-3B (ours)	69.9	71.4	-	-
T5-3B+PICARD	74.1	76.3	-	-
T5-3B†	71.5	74.4	68.0	70.1
T5-3B+PICARD†	**75.5**	**79.3**	**71.9**	**75.1**

Table 1: Our results (bottom) and relevant prior art (top) on the Spider text-to-SQL task. Shown are the exact-set-match accuracy (EM) and execution accuracy (EX) percentages on Spider's development and test sets. Our results are for a beam of size 4, and PICARD is parsing with guards for the top-2 token predictions. A dagger (†) indicates use of database content, otherwise schema only.

System	Development		Test	
	QM%	IM%	QM%	IM%
RATSQL + SCoRe (Yu et al., 2021)	52.1	22.0	51.6	21.2
T5-3B	53.8	21.8	51.4	21.7
T5-3B+PICARD	**56.9**	**24.2**	**54.6**	**23.7**

Table 2: Our results (bottom) and relevant prior art (top) on the CoSQL dialog state tracking task. Shown are the question match accuracy (QM) and interaction match accuracy (IM) percentages on CoSQL's development and test sets. Our results are for a beam of size 4, and PICARD is parsing with guards for the top-2 token predictions.

et al. (2020), we have studied also what happens when PICARD is only checking hypotheses when the model predicts their finalization with the end-of-sequence token.[2] In this restrained mode, PICARD is still effective, but much less so compared to normal incremental operation. The gap between these two modes of operation only begins to shrink for large beam sizes. This is understandable since Lin et al. (2020) used beam sizes of at least 16 and up to 64 to reach optimal results with filtering while Suhr et al. (2020) used a beam of size 100.

4 Conclusion

We propose and evaluate a new method, PICARD, for simple and effective constrained decoding with large pre-trained language models. On both, the Spider cross-domain and cross-database text-to-SQL dataset and the CoSQL SQL-grounded dialog state tracking dataset, we find that the PICARD decoding method not only significantly improves the performance of fine-tuned but otherwise unmodified T5 models, it also lifts a T5-3B model to state-of-the-art results on the established exact-match and execution accuracy metrics.

[2]This is not exactly equivalent to filtering a completely finalized beam, because the hypotheses rejected by PICARD never enter it and never take up any space.

Acknowledgements

We thank Lee Zamparo for his contributions to the experiments on the CoSQL dataset. Further, we would like to thank Pete Shaw for his input on the reproduction of the T5 results on Spider. We would also like to extend our gratitude to Tao Yu and Yusen Zhang for their efforts in evaluating our model on the test split of the Spider and CoSQL datasets. Finally, we thank our anonymous reviewers for their time and valuable suggestions.

References

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: Line graph enhanced text-to-SQL model with mixed local and non-local relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2541–2555, Online. Association for Computational Linguistics.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. RYANSQL: Recursively Applying Sketch-based Slot Fillings for Complex Text-to-SQL in Cross-Domain Databases. *Computational Linguistics*, 47(2):309–332.

Daan Leijen and Erik Meijer. 2001. Parsec: Direct style monadic parser combinators for the real world. Technical Report UU-CS-2001-27. User Modeling 2007, 11th International Conference, UM 2007, Corfu, Greece, June 25-29, 2007.

Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. 2019. Grammar-based neural text-to-sql generation.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Bryan O'Sullivan and Ben Gamari. 2021. attoparsec: Fast combinator parsing for bytestrings and text. Software available on the Haskell package repository.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Ohad Rubin and Jonathan Berant. 2021. SmBoP: Semi-autoregressive bottom-up semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 311–324, Online. Association for Computational Linguistics.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 922–938, Online. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cicero Nogueira dos Santos, and Bing Xiang. 2021. Learning contextual representations for semantic parsing with generation-augmented pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13806–13814.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8372–8388, Online. Association for Computational Linguistics.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Peng Xu, Dhruv Kumar, Wei Yang, Wenjie Zi, Keyi Tang, Chenyang Huang, Jackie Chi Kit Cheung, Simon J.D. Prince, and Yanshuai Cao. 2021. Optimizing deeper transformers on small datasets. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2089–2102, Online. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2018. Tranx: A transition-based neural abstract syntax parser for semantic parsing and code generation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2021. Score: Pre-training for context representation in conversational semantic parsing. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-sql with distilled test suites. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.