DATABASE-AWARE ASR ERROR CORRECTION FOR SPEECH-TO-SQL PARSING

Yutong Shao, Arun Kumar, Ndapa Nakashole

University of California, San Diego
Computer Science and Engineering
9500 Gilman Drive, La Jolla, CA 92093

ABSTRACT

We study the task of spoken natural language to SQL parsing (speech-to-SQL), where the goal is to map a spoken utterance to the corresponding SQL. A simple way to develop a speech-to-SQL parser is to pass the speech to an automatic speech recognition (ASR) system, and pass the transcription to a text-to-SQL parser. However, ASR is still error-prone. We propose an ASR correction method, DBATI (DataBase-Aware TaggerILM). The method first detects erroneous spans in the input, and rewrites each span. Our method leverages a novel joint representation of text and the database (DB). Our experiments show that our method yields better performance on both text quality and downstream SQL accuracy, compared to existing ASR error correction methods.

***Index Terms*—** Speech-to-SQL, ASR error correction, database, natural language interface

1. INTRODUCTION

Interfaces that support human language as a medium of communication between humans and computers have been of interest for decades [1, 2, 3]. Known as Natural Language Interfaces (NLIs), early systems saw limited success due to the difficult problem of endowing computers with the ability to understand natural language. Progress in language understanding has led to renewed interest in NLIs. In particular, several studies have focused on NLIs to databases (NLIDBs) [4, 5, 6]. NLIDBs, when fully realized, stand to support users who are not proficient in query languages. The primary focus of NLIDBs has been on parsing natural language *text* utterances into executable SQL queries (text-to-SQL parsing). Motivated by the rise of speech-driven digital assistants on smartphones, tablets, and other small handheld devices, we study the task of parsing spoken natural language to executable SQL queries (speech-to-SQL parsing).

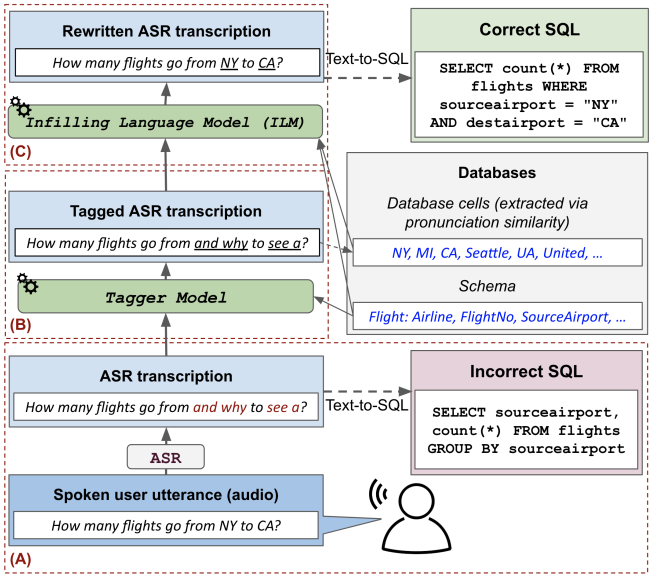

Fig. 1: Overview of the speech-to-SQL task and our method. (A) Directly passing the erroneous ASR transcription to text-to-SQL parser will likely produce wrong SQL output. We apply a Tagger (B) and an ILM rewriter (C) to fix the transcription, incorporating ASR transcription and DB information, in order to obtain the correct user utterance.

Motivation. A speech-to-SQL parser has a number of potential use cases. For example, in the healthcare domain, a nurse practitioner at a patient bedside typically looks up patient details on a desktop in the patient's room by filling out forms whose back-end is a database. In such a scenario, speech-to-SQL could be used instead, for faster results. Furthermore, speech-to-SQL removes the need for keyboards that can be slow and cumbersome on small devices, when querying databases.

Gold	Whose name has substring ABC ?
ASR	Who's name has a sub string ABC .
Tags	U-EDIT KEEP KEEP U-DEL B-EDIT L-EDIT KEEP U-EDIT
ILM Input	*Who's name has sub string ABC .*
ILM Pred.	*Whose [ANS] substring [ANS] ? [ANS]*
Final Output	*Whose name has substring ABC ?*

Table 1: Infilling Language Model (ILM) example. According to the tags, *"Who's"*, *"sub string"* and *"."* are EDIT spans (shown as underlined) and *"a"* is a DEL span.

Approach. To build a speech-to-SQL parser, we can leverage progress in text-to-SQL parsing, and automatic speech recognition (ASR). However, ASR is still error-prone. To deal with ASR errors, we propose an error correction method that fixes ASR errors in the context of the DB. Our error correction method, *DBATI* (DataBase-Aware TaggerILM), edits the ASR transcription by tagging tokens to indicate if they should be edited (Tagger), and then rewriting the appropriate tokens using an Infilling language model (ILM). We build both the Tagger and ILM on top of a novel joint representation of text and DB schema. Figure 1 illustrates the task and our approach.

Summary of Contributions. We make the following contributions: i) bring attention to the practical problem of ASR error correction for speech-to-SQL parsing. ii) propose an ASR error correction method, DBATI, with a novel joint encoder of text and DB and outperforms other ASR error correction methods. iii) present a new dataset which is a spoken version of the *Spider* text-to-SQL benchmark [5], named *Spoken Spider*. We will make all our data available for reproducing our experiments, and to facilitate future research on this important but under-studied problem.

2. METHOD

Our method, DBATI, consists of two components: a *Tagger* and an *Infilling Language Model (ILM) rewriter*. The input to the two components includes the potentially erroneous ASR transcription and the DB schema. The desired output is a corrected ASR transcription.

Tagging Tokens. The *Tagger* aims to classify errors in the ASR transcription. It assigns one of three tags to each token in the input transcription: KEEP, DEL or EDIT. These tags are denoted as *rewriter tags*. The DEL and EDIT tags are based on the BILOU tagging schema, thus marking certain *spans* in the sentence to be deleted or edited[1]. See Table 1 for a example.

Rewriting Spans. After tagging, the *ILM rewriter* predicts the text rewrites for each EDIT span. The rewriter is an Infilling Language Model (ILM) [7], which is an autoregressive generation model for *blank filling*. A working example of the ILM rewriter is shown in Table 1. It takes a tagged sentence, and predicts a sequence with new text for each EDIT span. Text for different spans is separated by a special [ANS] token.

In what follows, we describe the technical details of the Tagger and the ILM rewriter. They share a joint encoder, but have separate decoders. The encoder fuses free text and structured DB representations.

2.1. Joint Encoder

Database Cells Extraction. As a pre-processing step, we extract cells from the DB that could be relevant for the correction task. Conceptually, cell values are can be useful because they tend to be domain-specific values that can appear in utterances but might not be in general-domain text, therefore often wrongly transcribed. However, the number of cells in a DB can be very large. It is impractical to add all cells to the input, thus we propose a cell extraction step. We select cells with the highest *pronunciation similarities* with the tagged EDIT spans. This step is only applied in the *ILM rewriter* step when EDIT spans are available.

The pronunciations are represented by *phoneme* sequences. The phonemes of each utterance token or DB cell token are obtained from the CMU pronunciation dictionary[2]; if not found, they are inferred from Espeak[3]. We compute the similarity of phoneme sequences using $s(p_1, p_2) = 1 - \frac{d(p_1,p_2)}{\max(|p_1|,|p_2|)}$ where d is the edit-distance and $|p_i|$ denotes the length of p_i. We select K cells from the DB with highest similarities with each EDIT span, a total of $K \times E$ cells where E is the number of EDIT spans in the input. The selected cells are then added to the *DB token sequence*. The DB token sequence includes tables and columns from the DB schema; for the ILM rewriter, the extracted cell values are also included. The DB token sequence is transformed using DB *serialization*, and takes the form: *"Table1 : col1 (cell1, cell2, ...),*

[1] BILOU: Begin, Inside, Last, Outside, Unit. Tokens with tags from B-DEL to L-DEL, or a single U-DEL, make a DEL span; same for EDIT spans. The KEEP tags are O tags for DEL and EDIT spans.

[2] https://github.com/cmusphinx/cmudict

[3] http://espeak.sourceforge.net

col2 (cell1, cell2, ...), ... ; Table2 : col1 ...", following previous work [8].

PLM Encoding of Text + DB. We encode both the text ASR transcription and the DB token sequence using a pre-trained language model (PLM). We chose a BART encoder because its pre-training objective involves denoising, which is relevant to our task. We concatenate the text with the DB token sequence and feed the sequence into the PLM to get a contextualized text-DB encoding for each token (using a learnable scalar mix of the hidden representations from each encoder layer).

Other features. We use *align tags* which mark whether all ASR candidates agree on a token, providing additional information for token correctness. Futhermore, the ILM rewriter, the *rewriter tags* are utilized as input features. These features are vector-embedded using a look-up embedding table.

Joint Representation. We concatenate the PLM encodings and tag embeddings of each token, and feed the result into a self-attention layer and a standard LSTM sequence encoder to get the joint representation. The joint representation is then used by the decoder of either the Tagger or the rewriter.

2.2. Decoders

Tagger Decoder. The Tagger decoder is an LSTM-CRF sequence labeler. During training, we require "gold rewriter tags" to supervise the Tagger. We leverage work on aligning tokens in machine translation. Specifically, we apply Fast Aligner [9] to every ASR candidate and its corresponding clean text, and obtain the "gold rewriter tags" for each token based on the output alignment.

Infilling LM Decoder. The ILM rewriter decoder is a standard LSTM-based decoder with a copy mechanism [10]. The target output is the sequence of tokens that fill in the blanks as shown in Table 1.

3. EXPERIMENTS

Our baselines and proposed methods make use of an external ASR system and text-to-SQL system. For ASR, we use Amazon Transcribe, which is a state-of-the-art commercial ASR system whose outputs includes the transcription candidate list. It also outputs the timestamps and confidence scores of each transcribed token in each candidate. For text-to-SQL parsing, our experiments use three representative parsers: RAT-SQL [6], T5-base, and T5-large [11, 12].

Dataset. The main dataset we use is Spider [5]. Spider is a large-scale text-to-SQL dataset in which the train, dev, and test data are different subsets of the DB, thus models must generalize to unseen DBs. In order to evaluate speech-to-SQL systems, we created a spoken version of Spider, named Spoken Spider. We used Amazon Polly text-to-speech (TTS) synthesizer to obtain the audio of all natural language queries. We also collected real human speech on a subset of test samples; the details are described in section 3.1.

Evaluation Metrics. For evaluation, we use Word Error Rate (WER) and BLEU score as metrics for text quality. To measure the end-to-end speech-to-SQL performance, we use the SQL *exact match* and *execution match* score provided in the Spider official evaluation script, corresponding to the Spider leaderboards.

Baselines. We compare to the following baseline methods for ASR error correction. To mirror the settings in previous work [13, 14], for all baselines, we use our encoder without DB tokens and other features.

1) Blackbox. This baseline applies ASR to the spoken query, and passes the transcription to a text-to-SQL parser. No ASR error correction is performed.

2) Re-ranking Methods. These methods re-rank the N-best candidates from ASR system [13]. No DB information is considered by these methods, and the output is limited to be one of the ASR transcription candidates. For our *Reranker* baseline, the "decoder" is a multi-layer perceptron (MLP) predicting the probability of an ASR candidate to be the best one among all candidates.

3) Rewriting Methods. Rewriting methods aim to directly reconstruct the correct text from the ASR transcription. Previous work applies a Seq2seq (S2S) model to map ASR output directly into the full sentence of corrected text, without tagging and ILM as we have [14] and no DB information is considered. For our *S2S-rewriter* baseline, we use the same decoder architecture as our ILM rewriter decoder, but removing the *rewriter tags* and use the full sentence as target output.

Evaluation Results. The main results are shown in Table 2. We trained each model at least 4 times, only varying the random seeds between runs, and report the mean and standard deviations of each metric.

All three methods substantially improve the text quality (on WER and BLEU) upon the baseline which does

Method	WER(↓)	BLEU(↑)	RAT-SQL		T5-base		T5-large	
			Exact(↑)	Exec	Exact(↑)	Exec(↑)	Exact(↑)	Exec(↑)
Blackbox	0.1194	0.8010	0.4552	n/a	0.4570	0.4570	0.5265	0.5119
Reranker	$0.1029_{\pm 0.0017}$	$0.8163_{\pm 0.0022}$	$0.4859_{\pm 0.0046}$	n/a	$0.4859_{\pm 0.0041}$	$\mathbf{\textit{0.4900}}_{\pm 0.0058}$	$0.5370_{\pm 0.0087}$	$0.5393_{\pm 0.0054}$
S2S-rewriter	$0.0912_{\pm 0.0051}$	$0.8350_{\pm 0.0055}$	$0.4858_{\pm 0.0135}$	n/a	$0.4584_{\pm 0.0085}$	$0.4470_{\pm 0.0144}$	$0.5407_{\pm 0.0157}$	$0.5018_{\pm 0.0116}$
TI (TaggerILM w/o JE.)	$\mathbf{\textit{0.0689}}_{\pm 0.0050}$	$\mathbf{\textit{0.8725}}_{\pm 0.0093}$	$\mathbf{\textit{0.5270}}_{\pm 0.0097}$	n/a	$\mathbf{\textit{0.4927}}_{\pm 0.0106}$	$\mathbf{\textit{0.4877}}_{\pm 0.0137}$	$0.5786_{\pm 0.0170}$	$0.5681_{\pm 0.0148}$
DBATI (TaggerILM w/ JE.)	$\mathbf{0.0651}_{\pm 0.0051}$	$\mathbf{0.8778}_{\pm 0.0092}$	$\mathbf{0.5393}_{\pm 0.0144}$	n/a	$\mathbf{0.5018}_{\pm 0.0097}$	$\mathbf{0.4913}_{\pm 0.0118}$	$\mathbf{0.5950}_{\pm 0.0116}$	$\mathbf{0.5859}_{\pm 0.0150}$
Gold query	0.0000	1.0000	0.6234	n/a	0.5832	0.6033	0.6746	0.6929

Table 2: Main results on Spoken Spider (Spider with TTS-generated speech). *JE* denotes the joint encoder structure we propose in this work for TaggerILM. The *w/o JE.* corresponds to settings in previous work. In **bold** are the best results; in ***italic*** are the results within the range of $1\times$ standard deviation from the best results. RAT-SQL predictions do not include value literals, thus we do not report its execution match.

not address ASR errors, consistent with findings in previous work. On SQL accuracy, all the methods also yield a large performance gain, except for S2S-rewriter which has no clear improvement when using T5 parsers. Among all methods, our DBATI achieves the overall best performance, achieving the highest score on all metrics. Without the joint encoder incorporating DB information, the TaggerILM framework still outperforms other baselines. However, performance is slightly or significantly below DBATI on different metrics. Lastly, there is large performance gap between all methods and directly passing the gold queries to text-to-SQL parser. This shows room for further improvement on the task of ASR correction for speech-to-SQL.

3.1. Human Speech Test

To test the domain-transfer capabilities of each method from Text-to-Speech (TTS) data to the real world setting, we sampled a subset of 100 queries from our test set and collected spoken queries from a human speaker on each sample. Evaluation results are shown in Table 3. First, performance of all methods dropped compared to the TTS-synthesized data, showing that human speech is indeed noisier and challenging. Compared to baseline methods, DBATI still improves the text quality and SQL accuracy with T5-large parser, and achieves comparable performance with RAT-SQL and T5-base. Given that T5-large is the best parser in our study, having outstanding performances on this parser is promising. Overall, our model trained on TTS data maintains a very strong performance on real data in comparison to baselines.

4. RELATED WORK

While there is previous work on speech-to-SQL [15, 16, 17], to our knowledge, our work is the first to systematically explore neural ASR error correction approaches for SQL-related tasks. On ASR correction, previous work proposed re-ranking methods [13, 18] and S2S rewriting methods [14]. Recently, contemporaneous work proposed an ASR correction framework based on error region detection and per-region correction, conceptually similar to our TaggerILM framework [19]. The main difference is that we focus on a task involving structured data, speech-to-SQL, and propose a joint text-DB encoding that provides input to our TaggerILM framework.

Method	WER	Exact (RAT-SQL)	Exec (T5-base)	Exec (T5-large)
Blackbox	0.1733	0.3500	**0.4200**	0.4600
Reranker	$0.1689_{\pm 0.0055}$	$0.3725_{\pm 0.0096}$	$\mathbf{\textit{0.4075}}_{\pm 0.0236}$	$0.4775_{\pm 0.0222}$
S2S-rewriter	$0.1427_{\pm 0.0046}$	$\mathbf{\textit{0.3950}}_{\pm 0.0404}$	$\mathbf{\textit{0.4050}}_{\pm 0.0252}$	$0.4625_{\pm 0.0310}$
TI	$0.1347_{\pm 0.0020}$	$\mathbf{0.4175}_{\pm 0.0263}$	$\mathbf{\textit{0.3925}}_{\pm 0.0377}$	$0.4625_{\pm 0.0310}$
DBATI	$\mathbf{0.1294}_{\pm 0.0029}$	$\mathbf{\textit{0.4125}}_{\pm 0.0171}$	$\mathbf{\textit{0.4100}}_{\pm 0.0216}$	$\mathbf{0.5075}_{\pm 0.0222}$
Gold query	0.0000	0.5600	0.6000	0.7100

Table 3: Domain-transfer evaluation results on human speech. Due to space limit, we only show a subset of representative metrics.

5. CONCLUSION

We proposed an ASR error correction method for speech-to-SQL parsing. Our proposed method, DBATI, is able to more effectively fix ASR errors compared to baselines, as reflected by superior performances on both text and SQL accuracy on several different text-to-SQL parsers and in different domains (TTS-synthesized speech / human speech). Future work can explore techniques in leveraging audio information in ASR correction, such as better modal-fusion mechanisms. Another future direction is to investigate transfer learning strategies to improve model performance in real-world scenarios where the data are more noisy and dissimilar to training data.

6. REFERENCES

[1] Terry Winograd, "Procedures as a representation for data in a computer program for understanding natural language," Tech. Rep., Massachusetts Institute of Technology, 1971.

[2] John M Zelle and Raymond J Mooney, "Learning to parse database queries using inductive logic programming," in *Proceedings of the national conference on artificial intelligence*, 1996, pp. 1050–1055.

[3] Luke S Zettlemoyer and Michael Collins, "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars," *arXiv preprint arXiv:1207.1420*, 2012.

[4] Victor Zhong, Caiming Xiong, and Richard Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[5] Tao Yu, Rui Zhang, Kai Yang, et al., "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," in *EMNLP*, 2018, pp. 3911–3921.

[6] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson, "RAT-SQL: relation-aware schema encoding and linking for text-to-sql parsers," in *ACL*, 2020, pp. 7567–7578.

[7] Chris Donahue, Mina Lee, and Percy Liang, "Enabling language models to fill in the blanks," in *ACL*. 2020, pp. 2492–2501, Association for Computational Linguistics.

[8] Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova, "Compositional generalization and natural language variation: Can a semantic parsing approach handle both?," *ArXiv*, vol. abs/2010.12725, 2021.

[9] Chris Dyer, Victor Chahuneau, and Noah A. Smith, "A simple, fast, and effective reparameterization of ibm model 2," in *HLT-NAACL*, 2013.

[10] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li, "Incorporating copying mechanism in sequence-to-sequence learning," *ArXiv*, vol. abs/1603.06393, 2016.

[11] Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *ArXiv*, vol. abs/1910.10683, 2020.

[12] Tianbao Xie, Chen Henry Wu, Peng Shi, et al., "Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models," 2022.

[13] Yue Weng, Sai Sumanth Miryala, Chandra Khatri, Runze Wang, Huaixiu Zheng, Piero Molino, M. Namazifar, A. Papangelis, H. Williams, Franziska Bell, and G. Tur, "Joint contextual modeling for asr correction and language understanding," *ICASSP 2020*, pp. 6349–6353, 2020.

[14] Anirudh Mani, Shruti Palaskar, Nimshi Venkat Meripo, Sandeep Konam, and F. Metze, "Asr error correction and domain adaptation using machine translation," *ICASSP 2020*, pp. 6344–6348, 2020.

[15] S. Jamoussi, Kamel Smaïli, and J. Haton, "From speech to sql queries : a speech understanding system," in *AAAI 2005*, 2005.

[16] S. Kumar, A. Kumar, P. Mitra, and G. Sundaram, "System and methods for converting speech to sql," *ArXiv*, vol. abs/1308.3106, 2013.

[17] Shravankumar Hiregoudar, Manjunath Gonal, and Karibasappa K G, "Speech to sql generator-a voice based approach," *Journal of Basic and Applied Research International*, vol. Vol 4, pp. 01–05, 01 2019.

[18] Rodolfo Corona, Jesse Thomason, and R. Mooney, "Improving black-box speech recognition using semantic parsing," in *IJCNLP 2017*, 2017.

[19] Fan Zhang, Mei Tu, Song Liu, and Jinyao Yan, "Asr error correction with dual-channel self-supervised learning," *ICASSP 2022*, pp. 7282–7286, 2022.