A. MODEL ILLUSTRATIONS

Figure 1 illustrates the full structure of our Joint Encoder. Figure 2 shows the DB cell extraction step.

B. SPOKEN SPIDER DATASET DETAILS

In order to evaluate speech-to-SQL systems, we created a spoken version of Spider, named Spoken Spider. We provide the details of this dataset here.

In the original Spider dataset, each sample contains a text query, a database ID and target SQL. We pass the text query to Amazon Polly text-to-speech (TTS) synthesizer[1] to get the synthesized speech audio. For a subset of test samples, we also collected human speech by recording an author of the paper reading the text queries. For any type of speech, we transcribe it using Amazon ASR service[2] and add the raw transcriptions to the sample as well. The ASR system returns top-K confident transcriptions instead of only the top-1, and we include them all into our dataset. As a result, one original Spider example may generate several Spoken Spider examples. Our method, as an error correction method, is trained on samples with synthesized speech transcriptions, and tested on both synthesized and human-read samples. Besides, we split the original validation set of Spider into two subsets of samples, one for validation and one for testing (since the original Spider test set is not publicly available). We follow the original guideline of splitting samples by database ID, i.e. the validation samples and test samples do not share any databases. Statistics of the dataset is shown in Table 1.

Dataset splits	# of clean queries	# of ASR candidate queries
Training	7000	41112
Dev	487	2707
Test	547	3075

Table 1: Spoken Spider statistics (synthetic speech data). The training set comes from "train_spider.json" in original Spider; the dev and test set come from "dev.json". For human speech, we sample 100 from 547 Spider examples from the test set and use the corresponding Spoken Spider samples to obtain the results.

[1]`https://aws.amazon.com/polly`
[2]`https://aws.amazon.com/transcribe`

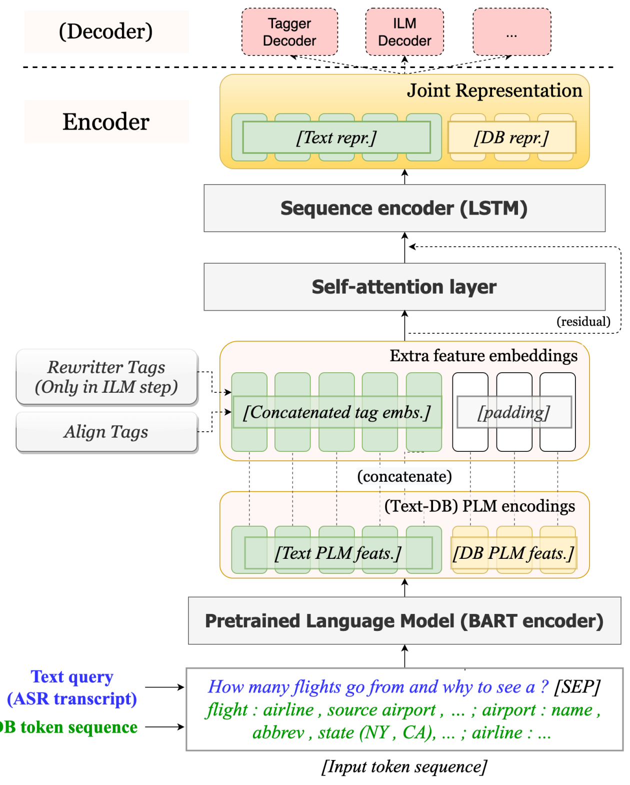

Fig. 1: Illustration of the major part of our model (the text-DB joint encoder).

C. EXTRA RESULTS

C.1. More Results on Main Experiments

We added BLEU as another word-level metrics. We also added more combinations of backbone models and types of encoders; especially, we added configurations with previous model, Reranker and S2S-rewriter, together with our proposed Joint Encoder (JE), to further profile the source of performance gain. The results are in Table 2 and 3. We notice that the gain mainly comes from TaggerILM. The Joint Encoder is able to further improve the TaggerILM, and provides an overall improvement for previous methods as well (except for S2S-rewriter on human speech where performance decreased).

C.2. Oracle Analysis

To examine the bottleneck of our method, we tested the performance of the Tagger and ILM rewriter separately, with the other part replaced by an oracle. In detail, the oracle Tagger always predicts the gold labels for rewriter tags; the oracle ILM rewriter always rewrites an `EDIT`

Method	WER	BLEU	RAT-SQL		T5-base		T5-large	
			Exact	Exec	Exact	Exec	Exact	Exec
Blackbox	0.1194	0.8010	0.4552	-	0.4570	0.4570	0.5265	0.5119
Reranker (w/o JE.)	0.1029$_{\pm0.0017}$	0.8163$_{\pm0.0022}$	0.4859$_{\pm0.0046}$	-	0.4859$_{\pm0.0041}$	***0.4900***$_{\pm0.0058}$	0.5370$_{\pm0.0087}$	0.5393$_{\pm0.0054}$
Reranker (w/ JE.)	0.0968$_{\pm0.0013}$	0.8278$_{\pm0.0015}$	0.4863$_{\pm0.0000}$	-	0.4881$_{\pm0.0067}$	**0.4913**$_{\pm0.0051}$	0.5425$_{\pm0.0108}$	0.5416$_{\pm0.0077}$
S2S-rewriter (w/o JE.)	0.0912$_{\pm0.0051}$	0.8350$_{\pm0.0055}$	0.4858$_{\pm0.0135}$	-	0.4584$_{\pm0.0085}$	0.4470$_{\pm0.0144}$	0.5407$_{\pm0.0157}$	0.5018$_{\pm0.0116}$
S2S-rewriter (w/ JE.)	0.0799$_{\pm0.0015}$	0.8579$_{\pm0.0014}$	0.4895$_{\pm0.0117}$	-	0.4886$_{\pm0.0038}$	0.4717$_{\pm0.0033}$	0.5453$_{\pm0.0135}$	0.5224$_{\pm0.0150}$
TaggerILM (w/o JE.)	***0.0689***$_{\pm0.0050}$	***0.8725***$_{\pm0.0093}$	***0.5270***$_{\pm0.0097}$	-	***0.4927***$_{\pm0.0106}$	***0.4877***$_{\pm0.0137}$	0.5786$_{\pm0.0170}$	0.5681$_{\pm0.0148}$
[DBATI] TaggerILM (w/ JE.)	**0.0651**$_{\pm0.0051}$	**0.8778**$_{\pm0.0092}$	**0.5393**$_{\pm0.0144}$	-	**0.5018**$_{\pm0.0097}$	**0.4913**$_{\pm0.0118}$	**0.5950**$_{\pm0.0116}$	**0.5859**$_{\pm0.0150}$
Gold text	0.0000	1.0000	0.6234	-	0.5832	0.6033	0.6746	0.6929

Table 2: Full results on Spoken Spider (Spider with TTS-generated speech). JE denotes the text-DB Joint Encoder. *TaggerILM (w/ JE.)* is our full DBATI method. 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× standard deviation from the best results. RAT-SQL predictions do not include value literals, thus we do not report its execution match.

Method	WER	BLEU	RAT-SQL		T5-base		T5-large	
			Exact	Exec	Exact	Exec	Exact	Exec
Blackbox	0.1733	0.6934	0.3500	-	***0.4000***	***0.4200***	0.4500	0.4600
Reranker (w/o JE.)	0.1689$_{\pm0.0055}$	0.6981$_{\pm0.0095}$	0.3725$_{\pm0.0096}$	-	0.3625$_{\pm0.0096}$	0.4075$_{\pm0.0236}$	0.4650$_{\pm0.0208}$	0.4775$_{\pm0.0222}$
Reranker (w/ JE.)	0.1586$_{\pm0.0010}$	0.7071$_{\pm0.0034}$	0.3775$_{\pm0.0126}$	-	**0.4025**$_{\pm0.0096}$	***0.4325***$_{\pm0.0096}$	0.4800$_{\pm0.0082}$	**0.5100**$_{\pm0.0115}$
S2S-rewriter (w/o JE.)	0.1427$_{\pm0.0046}$	0.7251$_{\pm0.0099}$	***0.3950***$_{\pm0.0404}$	-	***0.3950***$_{\pm0.0265}$	0.4050$_{\pm0.0252}$	***0.4925***$_{\pm0.0275}$	0.4625$_{\pm0.0310}$
S2S-rewriter (w/ JE.)	0.1581$_{\pm0.0061}$	0.7015$_{\pm0.0113}$	0.3875$_{\pm0.0171}$	-	***0.3850***$_{\pm0.0332}$	0.3925$_{\pm0.0096}$	***0.4850***$_{\pm0.0370}$	0.4750$_{\pm0.0252}$
TaggerILM (w/o JE.)	0.1347$_{\pm0.0020}$	0.7343$_{\pm0.0052}$	**0.4175**$_{\pm0.0263}$	-	***0.3825***$_{\pm0.0299}$	0.3925$_{\pm0.0377}$	0.4500$_{\pm0.0258}$	0.4625$_{\pm0.0310}$
[DBATI] TaggerILM (w/ JE.)	**0.1294**$_{\pm0.0029}$	**0.7456**$_{\pm0.0040}$	***0.4125***$_{\pm0.0171}$	-	***0.3900***$_{\pm0.0082}$	0.4100$_{\pm0.0216}$	**0.5125**$_{\pm0.0206}$	***0.5075***$_{\pm0.0222}$
Gold text	0.0000	1.0000	0.5600	-	0.5700	0.6000	0.6800	0.7100

Table 3: Full results for domain-transfer evaluation results on human speech.

Tagger	ILM	WER	Exact (RAT-SQL)	Exec (T5-base)	Exec (T5-large)
Trained	Trained	0.0666$_{\pm0.0035}$	0.5361$_{\pm0.0084}$	0.4895$_{\pm0.0111}$	0.5809$_{\pm0.0164}$
Oracle	Trained	0.0517$_{\pm0.0021}$	0.5366$_{\pm0.0071}$	0.4922$_{\pm0.0018}$	0.5731$_{\pm0.0081}$
Trained	Oracle	**0.0361**$_{\pm0.0021}$	**0.5755**$_{\pm0.0051}$	**0.5437**$_{\pm0.0049}$	**0.6446**$_{\pm0.0063}$

Table 4: Oracle analysis results showing that the ILM is the current performance bottleneck.

span with its aligned tokens in the gold utterance.

The results are in Table 4. Using an oracle tagger only yields small improvement on WER and no significant improvements for SQL. However, using an oracle ILM rewriter provides a significant performance boost on all metrics. We therefore conclude that ILM rewriter is the bottleneck in the TaggerILM framework and should be the focus of future work for further improvements.

C.3. Ablation Study

We did ablation study to skip the cell extraction step or completely remove the DB (we do the study on the ILM part only, as it is shown to be the performance bottleneck). The results in Table 5 show that in both settings, there is a clear performance drop. It is also worth noticing that, if we use DB but skip the cell extraction step (2nd row), the performance could be similar or even worse than not using the DB at all (3rd row), indicating the importance of the cell extraction step.

C.3.1. Syntactic Category Analysis

To better understand the strengths and limitations of our approach, we analyzed the token accuracy of rewritten utterances on each Part-of-Speech (POS) tag with >100 occurrences. The results are shown in Table 6, under meta-column "Token Accuracy". In detail, we use SpaCy to assign a POS tag to each token in the ASR transcription. We use a simple dynamic programming edit-distance algorithm to align the rewritten utterance to the gold one, check if each token correctly aligns with the rewritten query, and compute the percentage of correct tokens for each POS tag respectively. This percentage is used as *token accuracy*. We find that on most POS tags, our rewriter makes an improvement (column "Rewriter ΔAcc"), except for certain POS such as ADP (adpositions, such as "of", "in", etc.) and ADJ (adjectives) on which the ASR precision is already very high. The largest improvements are on PUNCT (punctuation), NUM (nu-

Ablation	Spoken Spider (TTS) WER	Spoken Spider (TTS) Exact (RAT-SQL)	Spoken Spider (TTS) Exec (T5-base)	Spoken Spider (TTS) Exec (T5-large)	Domain Transfer (Human Speech) WER	Domain Transfer (Human Speech) Exact (RAT-SQL)	Domain Transfer (Human Speech) Exec (T5-base)	Domain Transfer (Human Speech) Exec (T5-large)
Full model	0.0651$_{\pm 0.0051}$	0.5393$_{\pm 0.0144}$	0.4913$_{\pm 0.0118}$	0.5859$_{\pm 0.0150}$	0.1294$_{\pm 0.0029}$	0.4125$_{\pm 0.0171}$	0.4100$_{\pm 0.0216}$	0.5075$_{\pm 0.0222}$
- DB cells extraction	0.0689$_{\pm 0.0039}$	0.5302$_{\pm 0.0074}$	*0.4795$_{\pm 0.0105}$	*0.5617$_{\pm 0.0130}$	*0.1376$_{\pm 0.0045}$	0.3950$_{\pm 0.0252}$	*0.3800$_{\pm 0.0141}$	*0.4525$_{\pm 0.0150}$
- Full DB	0.0702$_{\pm 0.0054}$	0.5352$_{\pm 0.0069}$	*0.4758$_{\pm 0.0154}$	*0.5622$_{\pm 0.0189}$	*0.1355$_{\pm 0.0014}$	0.4200$_{\pm 0.0163}$	0.3900$_{\pm 0.0337}$	*0.4750$_{\pm 0.0208}$

Table 5: Ablation study (on ILM rewriter) for components in the multi-modal encoder. (*: performance drop by $> 1\times$ standard deviation, compared to the full model.)

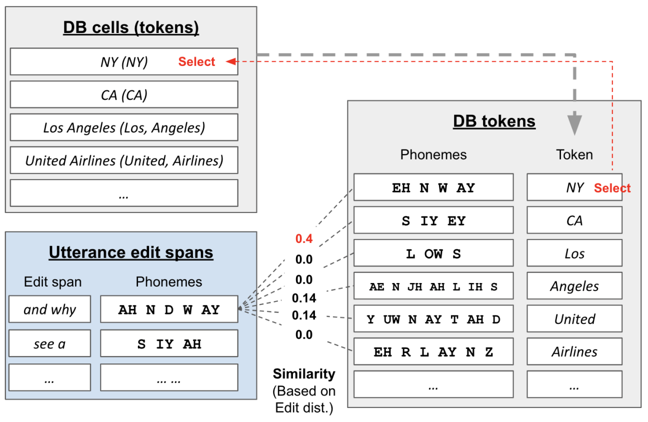

Fig. 2: Extracting DB cells with similar pronunciation phonemes to an `EDIT` span. For illustration purpose, we only show the process for `EDIT` span "and why", and we only select the top $K = 1$ similar cell. In our experiments, we use each `EDIT` span to select $K = 5$ similar cells and merge the selections.

merals) and CCONJ (conjunctions), which often involves prototypical and frequent ASR errors (e.g. "." vs. "?", "in" vs. "and", etc.) that are easily learned by the model. For PUNCT, many ASR outputs have periods at the end when the queries are questions. Also, some ASR outputs have extra ending punctuation marks in the middle. Errors on NUM are usually formatting errors, such as numbers transcribed into English words or mismatches in comma delimiters. Some of these errors actually appeared in samples in Table 7. Generally, the errors types listed above are more patterned and the rewriter is able to handle them well.

We also examine the influence of cell extraction by computing the token accuracy gain brought by cell extraction, i.e. compared to not using it. The results are shown in Table 6, column "Cell Ex. ΔAcc$\uparrow$". Without cell extraction, the rewritten token accuracy on **PROPN** would largely decrease and drop even below raw ASR; the accuracy on other POS are not much influenced. This finding confirms that cell extraction is beneficial for fixing proper nouns in ASR transcriptions.

To further examine the separate influence of each POS on the SQL exact match performance, we conducted a *freezing test* where we freeze tokens with certain POS during ILM rewriting. Conceptually, a higher performance loss when freezing a POS indicates that rewriting this POS is more helpful on the final performance[3]. We tested on all three backend parsers and show the results in Table 6, under meta-column "Freezing test". The most consistently contributing POS are **NOUN** and **ADP**. NOUN are important as expected, because most of the entity mentions that are directly related to SQL query are nouns. It is also the most frequency POS in Spider. Besides, ADP are also important because they often decide the relations of entities and thus crucial for SQL prediction (e.g. "of", "in", "with", etc.) Other influential POS are less explainable. Our assumption is that fixing these POS effectively maps the utterance back to the domain where text-to-SQL parser is trained, therefore improves performance.

Here we elaborate on an important detail: for token accuracy, the POS corresponds to tokens in the gold text (for a fair comparison of raw and rewritten text). However, for freezing test, the POS corresponds to tokens in the ASR transcription (by definition of freezing test). For example, for ADP, we notice that our rewriting is not helpful for the token accuracy, yet useful for SQL-related metrics. This is possibly because, many times ASR produces redundant or wrong ADP tokens that align to no token or non-ADP tokens in the gold text. These tokens are not counted toward ADP token accuracy; however, they count for the freeze test. Thus, these aforementioned results imply that the original ADP tokens in the gold text are already well-preserved by ASR; however, there are extra erroneous ADP tokens generated by ASR, and fixing these is important for SQL accuracy. The results of token accuracy and freezing test are not inconsistent,

[3]For freezing experiments, we only used the run with median performance (on exact match) among all runs. We treat it as a representative of all runs.

but complementary.

C.4. Sample Predictions

Several sample queries in which the TaggerILM corrected the query and improved the SQL predictions are shown in Table 7. The rewriter improves SQL accuracy by fixing critical ASR errors, such as “id” recognized as “idea”, “and” as “in” in 7a. It also fixes proper nouns such as “JetBlue Airways” as “check Blue Airways” in 7b. Although the “check” is not removed, which is an error (on the tagger side), it is useful enough to correct the SQL prediction. In future work, to be able to fix such errors, the rewriter will need a better context understanding and/or a better audio representation.

C.5. Backend NLIDB Adaptation

Given sufficient computational resources, we can consider applying *domain adaptation* to the backend text-to-SQL parser. That is, we can treat ASR transcription text-to-SQL as the target domain, and directly train the backend text-to-SQL parser on target domain data. We use the same dataset, our Spoken Spider, as the training data for backend parsers, making a fair comparison with our proposed method DBATI.

The results are shown in Table 8. On SQL accuracy, adaptation methods are overall comparable to our DBATI rewriter (on non-adapted parser). Despite the potential performance gain from domain adaptation, it has several clear drawbacks in practice, compared to a direct text rewriter. First, the text rewriters are *parser-agnostic*. Once trained, they can be deployed with any new parsers without further training. However, to apply domain adaptation we have to re-train the full parser model. This is a significant concern given that current SOTA parsers are usually very large models, such as T5-3B; re-training such models can be highly demanding on GPU memory and computational cost. Further, in practical usage scenarios of speech-based systems, users usually expect a correct text transcription to assure the system is functioning properly. Not fixing the text can hurt user’s trust of the system, even with slightly higher end-to-end performance. Therefore, we argue that rewriter methods are still worthy to study and develop.

POS tags (Frequency) in gold text	Token Accuracy				Freezing Test (↓)		
	Raw	Rewritten	Rewrite ΔAcc↑	Cell Ex. ΔAcc↑	Exact (RAT-SQL)	Exec (T5-base)	Exec (T5-large)
NOUN (1833)	0.9471	0.9669	0.0197	0.0026	-0.0365	-0.0220	-0.0457
ADP (801)	0.9888	0.9863	-0.0025	0.0009	-0.0128	-0.0092	-0.0183
PUNCT (742)	0.5512	0.7524	0.2012	-0.0030	-0.0201	0.0018	-0.0146
DET (1073)	0.9814	0.9865	0.0051	0.0000	-0.0036	-0.0018	-0.0110
PRON (335)	0.9075	0.9806	0.0731	-0.0037	-0.0128	-0.0037	-0.0092
PROPN (295)	0.7051	0.7475	0.0424	**0.0627**	-0.0073	0.0073	-0.0055
NUM (110)	0.6455	0.7864	0.1409	0.0045	-0.0091	-0.0018	-0.0037
VERB (508)	0.9331	0.9724	0.0393	0.0030	-0.0036	0.0018	-0.0037
AUX (537)	0.9125	0.9548	0.0423	-0.0023	-0.0018	0.0000	-0.0018
ADJ (487)	0.9856	0.9821	-0.0035	0.0031	0.0019	-0.0018	0.0000
CCONJ (217)	0.7972	0.9355	0.1383	-0.0011	0.0000	0.0018	0.0000
ADV (163)	0.9816	0.9970	0.0154	0.0015	0.0000	0.0018	0.0000

Table 6: TaggerILM performance analysis with POS tags. For token accuracy, POS is determined by tokens in gold text. "Raw" stands for raw ASR transcription, and "Rewritten" for utterances rewritten by our model. "Rewrite ΔAcc↑" is the performance gain from rewriting; "Cell Ex. ΔAcc↑" is the performance gain from adding cell extraction. In **green** are the largest improvements. *Freezing test* show performance loss on each metric when freezing a POS, i.e. not rewriting tokens with this POS. The POS is determined by the raw ASR transcription. In **blue** are the largest drops.

Gold	Find the id and weight of all pets whose age is older than 1.
ASR	find the idea in weight of all pets whose ages older than one.
ASR SQL	SELECT AVG(weight) FROM pets WHERE pet_age > 1 (0)
Rewritten	find the id and weight of all pets whose age is older than 1.
Rewritten SQL	SELECT petid, weight FROM pets WHERE pet_age > 1 (1)

(a) Fixing nouns and adpositions.

Gold	Which abbreviation corresponds to Jetblue Airways?
ASR	which abbreviation corresponds to check Blue Airways.
ASR SQL	SELECT abbreviation FROM airlines WHERE airline = "check blue Airways" (0)
Rewritten	which abbreviation corresponds to check " jetblue airways?
Rewritten SQL	SELECT abbreviation FROM airlines WHERE airline = "JetBlue Airways" (1)

(b) Fixing proper nouns and punctuation.

Table 7: Samples improved by our TaggerILM rewriter. Numbers in brackets after SQL queries indicate the correctness (1 for correct, 0 otherwise).

			RAT-SQL		T5-base	
Method	WER	BLEU	Exact	Exec	Exact	Exec
Blackbox (no adapt)	0.1194	0.8010	0.4552	n/a	0.4570	0.4570
Blackbox (adapted)	0.1194	0.8010	0.5247	n/a	0.4954	0.4845
DBATI	0.0666	0.8781	0.5361	n/a	0.4986	0.4895

Table 8: Adaptation re-training results. We do not run adaptation experiments for T5-large due to high computational resources demand.