On the Potential of Lexico-logical Alignments for Semantic Parsing to SQL Queries

Tianze Shi[*]
Cornell University
tianze@cs.cornell.edu

Chen Zhao[*]
University of Maryland
chenz@cs.umd.edu

Jordan Boyd-Graber
University of Maryland
jbg@umiacs.umd.edu

Hal Daumé III
Microsoft Research & University of Maryland
me@hal3.name

Lillian Lee
Cornell University
llee@cs.cornell.edu

Abstract

Large-scale semantic parsing datasets annotated with logical forms have enabled major advances in supervised approaches. But can richer supervision help even more? To explore the utility of fine-grained, lexical-level supervision, we introduce SQUALL, a dataset that enriches 11,276 WIKITABLEQUESTIONS English-language questions with manually created SQL equivalents plus alignments between SQL and question fragments. Our annotation enables new training possibilities for encoder-decoder models, including approaches from machine translation previously precluded by the absence of alignments. We propose and test two methods: (1) supervised attention; (2) adopting an auxiliary objective of disambiguating references in the input queries to table columns. In 5-fold cross validation, these strategies improve over strong baselines by 4.4% execution accuracy. Oracle experiments suggest that annotated alignments can support further accuracy gains of up to 23.9%.

1 Introduction

The availability of large-scale datasets pairing natural utterances with logical forms (Dahl et al., 1994; Wang et al., 2015; Zhong et al., 2017; Yu et al., 2018, *inter alia*) has enabled significant progress on supervised approaches to semantic parsing (Jia and Liang, 2016; Xiao et al., 2016; Dong and Lapata, 2016, 2018, *inter alia*). However, the provision of logical forms alone does not indicate important fine-grained relationships between individual words or phrases and logical form tokens. This is unfortunate because researchers have in fact hypothesized that the lack of such *alignment* information hampers progress in semantic parsing (Zhang et al., 2019, pg. 80).

[*]Equal contribution; listed in alphabetical order.

Table: Province of Alessandria

City (c1)	*Population (c2)*	*Area (km^2) (c3)*	...
Alessandria	94191	203.97	...
Casale Monferrato	36039	86.32	...
Novi Ligure	28581	54.22	...
Tortona	27476	99.29	...
Acqui Terme	20426	33.42	...

Question: ①How many ②cities have ③at least ④25,000 ⑤people?

Target Logical Form:

`SELECT ①count(②c1)① FROM w WHERE ⑤c2_number③>=④25000`

Answer: 4

Table: Bulgaria at the 1988 Winter Olympics

Athlete (c1)	*Total Time (c2)*	*Total Rank (c3)*	...
Stefan Shalamanov	1:52.37	23	...
Borislav Dimitrachkov	1:50.81	19	...
Petar Popangelov	1:46.34	16	...

Question: ①Who has ②the highest ③rank ?

Target Logical Form:

`SELECT ①c1 FROM w ②ORDER BY ③c3_number ②LIMIT 1`

Answer: Petar Popangelov

Figure 1: Two examples from SQUALL. The table-question-answer triplets come from WIKITABLEQUESTIONS. We provide the logical forms as SQL plus alignments between question and logical form. In the bottom example, for instance, "the highest" ↔ `ORDER BY` and `LIMIT 1`, as indicated by both matching highlight color (blue) and circled-number labels (②).

We address this lack by introducing SQUALL,[1] the first large-scale semantic-parsing dataset with manual lexical-to-logical alignments; and we investigate the potential accuracy boosts achievable from such alignments. The starting point for SQUALL is WIKITABLEQUESTIONS (WTQ; Pasupat and Liang, 2015), containing data tables, English questions regarding the tables, and table-based answers. We manually enrich the 11,276-instance subset of WTQ's training data that is translatable to SQL

[1]SQUALL = "SQL+QUestion pairs ALigned Lexically".

by providing expert annotations, consisting not only of target logical forms in SQL, but also labeled alignments between the input question tokens (e.g., "how many") and their corresponding SQL fragments (e.g., `COUNT(...)`). Figure 1 shows two SQUALL instances.

These new data enable training of encoder-decoder neural models that incorporates manual alignments. Consider the bottom example in Figure 1: A decoder can benefit from knowing that `ORDER BY ... LIMIT 1` comes from "the highest" (where rank 1 is best); and an encoder should match "who" with the "athlete" column even though the two strings have no overlapping tokens. We implement these ideas with two training strategies:

1. *Supervised attention* that guides models to produce attention weights mimicking human judgments during both encoding and decoding. Supervised attention has improved both alignment and translation quality in machine translation (Liu et al., 2016; Mi et al., 2016), but has only been applied in semantic parsing to heuristically generated alignments (Rabinovich et al., 2017) due to the lack of manual annotations.

2. *Column prediction* that infers which column in the data table a question fragment refers to.

Using BERT features, our models reach 54.1% execution accuracy on the WTQ test set, surpassing the previous weakly-supervised state-of-the-art 48.8% (where weak supervision means access to only the answer, not the logical form of the question). More germane to the issue of alignment utility, in 5-fold cross validation, our additional fine-grained supervision improves execution accuracy by 4.4% over models supervised with only logical forms; ablation studies indicate that mappings between question tokens and columns help the most. Additionally, we construct *oracle* models that have access to the full alignments during test time to show the unrealized *potential* for our data, seeing improvements of up to 23.9% absolute logical form accuracy.

Through annotation-cost and learning-curve analysis, we conclude that lexical alignments are cost-effective for training parsers: lexical alignments take less than half the time to annotate as a logical form does, and we can improve execution accuracy by 2.5 percentage points by aligning merely 5% of the logical forms in the training set.

Our contributions are threefold: 1) we release a high-quality semantic parsing dataset with manually-annotated logical forms; 2) we label the alignments between the English questions and the corresponding logical forms to provide additional supervision; 3) we propose two training strategies that use our alignments to improve strong base models. Our dataset and code are publicly available at `https://www.github.com/tzshi/squall`.

2 Task: Table-based Semantic Parsing

Our task is to answer questions about structured tables through semantic parsing to logical forms (LFs). Formally, the input $x = (q, T)$ consists of a question q about a table T, and the goal of a semantic parser is to reproduce the target LF $y^\star$ for q (and thus have high *LF accuracy*) or, in a less strict setting, to generate any query LF y' that, when executed against T, yields the correct output $z^\star$ (and thus have high *execution accuracy*).

In a *weakly supervised* setting, training examples consist only of input-answer pairs $(x, z^\star)$. Recent datasets (Zhong et al., 2017; Yu et al., 2018, *inter alia*) provide enough logical forms, i.e., $(x, y^\star)$ training pairs, to learn from mappings from x to $y^\star$ in a *supervised* setting. Unsurprisingly, supervised models are more accurate than weakly supervised ones. However, training supervised models is still challenging: both x and y are structured, so models typically generate y in multiple steps, but the training data cannot reveal which parts of x generate which parts of y and how they are combined.

Just as adding supervised training improves accuracy over weak supervision, we explore whether even *finer*-grained supervision further helps. Since no large-scale datasets furnishing fine-grained supervision exist (to the best of our knowledge), we introduce SQUALL.

3 SQUALL: Our New Dataset

SQUALL is based on WIKITABLEQUESTIONS (WTQ; Pasupat and Liang, 2015). WTQ is a large-scale question-answering dataset that contains diverse and challenging crowd-sourced question-answer pairs over 2,108 semi-structured Wikipedia tables. Most of the questions are more than simple table-cell look-ups and are highly compositional, a fact that motivated us to study lexical mappings between questions and logical forms. We hand-generate SQL equivalents of the WTQ queries and align question tokens with corresponding SQL query fragments.[2] We leave lexical alignments of other text-to-SQL datasets and cross-dataset model generalization (Suhr et al., 2020) to future work.

3.1 Data Annotation

We annotated WTQ's training fold in three stages: database construction, SQL query annotation, and alignment. Two expert annotators familiar with SQL annotated half of the dataset each and then checked each other's annotations and resolved all conflicts via discussion. See Appendix C for the annotation guidelines.

Database Construction Tables encode semi-structured information. Each table column usually contains data of the same type: e.g., text, numbers, dates, etc., as is typical in relational databases. While pre-processing the WTQ tables, we considered both basic data types (e.g., raw text, numbers) and composite types (e.g., lists, binary tuples), and we suffixed column names with their inferred data types (e.g., `_number` in Figure 1). For annotation consistency, all tables were assigned the same name `w` and columns were given the sequential names `c1`, `c2`,... in the database schema, but we kept the original table headers for feature extraction. We additionally added a special column `id` to every table denoting the linear order of its rows. See Appendix D for details.

Conversion of Queries to SQL For every question in WTQ's training fold, we manually created its corresponding SQL query, choosing the shortest when there are multiple possibilities, for instance, we wrote "`SELECT MAX(c1) FROM w`" instead of "`SELECT c1 FROM w ORDER BY c1 DESC LIMIT 1`". An exception is that we opted for less table structure-dependent versions even if their complexity was higher. As an example, if the table listed games (`c2`) pre-sorted by date (`c1`), and the question was "what is the next game after A?", we wrote "`SELECT c2 FROM w WHERE c1 > (SELECT c1 FROM w WHERE c2 = A) ORDER BY c1 LIMIT 1`" instead of "`SELECT c2 FROM w WHERE id = (SELECT id FROM w WHERE c2 = A) + 1`". Out of 14,149 questions spanning 1,679 tables,

[2] SQL is a widely adopted formalism. Other formalisms including LambdaDCS (Pasupat and Liang, 2015), have been used on WTQ. SQL and LambdaDCS can express roughly the same percentage of queries: 81% (our finding) vs. 79% (analysis of a 200-question sample by Pasupat and Liang, 2016). We leave automatic conversion to and from SQL to other formalisms and vice versa to future work.

	how long	`MAX(...)`
Frequently aligned to	`col`	the last
	`MAX(col)-MIN(col)`	the most
	`col-col`	the largest
	`COUNT(*)`	the highest
	`COUNT(col)`	the first

Table 1: Examples of frequently-aligned English/LF segment pairs, illustrating the diversity in the aligned counterparts for the same lexical units. `col` is a placeholder for the actual data table column mention.

SQUALL provided SQL queries for 11,468 questions, or 81.1%. The remaining 18.9% consisted of questions with non-deterministic answers (e.g., "show me an example of ..."), questions requiring additional pre-processing (e.g., looking up a date inside a text-based details column), and cases where SQL queries would be insufficiently expressive (e.g., "what team has the most consecutive wins?").

Alignment Annotation Given a tokenized question/LF pair, the annotators selected and aligned corresponding fragments from the two sides. The selected tokens did note need to be contiguous, but they had to be units that decompose no further. For the example in Figure 1, there were three alignment pairs, where the non-contiguous "`ORDER BY ... LIMIT 1`" was treated as an atomic unit and aligned to "the highest" in the input. Additionally, not all tokens on either side needed to be aligned. For instance, SQL keywords `SELECT`, `FROM` and question tokens "what", "is", etc. were mostly unaligned. Table 1 shows that the same question phrase was aligned to a range of SQL expressions, and vice versa. Overall, 49.8% of question tokens were aligned. Comparative and superlative question tokens were the most frequently aligned, while many function words were unaligned; see Appendix E for part-of-speech distributions of the aligned and unaligned tokens. Except for the four keywords in the basic structure "`SELECT ... FROM w WHERE ...`", 90.2% of SQL keywords were aligned. The rest of the unaligned SQL tokens include d= (alignment ratio of 18.0%), `AND` (25.5%) and column names (86.1%). The first two cases arose because equality checks and conjunctions of filtering conditions are often implicit in natural language.

Inter-Annotator Agreement and Annotation Cost The two annotators' initial SQL annotation agreement in a pilot trial[3] was 70.4% and after discussion, they agreed on 94.5% of data instances; similarly, alignment agreement rose from 75.1% to 93.3%. With respect to annotation speed, an average SQL query took 33.9 seconds to produce and an additional 15.0 seconds to enrich with alignments: the cost of annotating 100 instances with alignment enrichment was comparable to that of 144 instances with only logical forms.

3.2 Post-processing

Literal values in the SQL queries such as "25,000" in Figure 1 and "star one" in Figure 3 are often directly copied from the input questions. We thus adapted WikiSQL's (Zhong et al., 2017) task setting, where all literal values correspond to spans in the input questions. We used our alignment to generate gold selection spans, filtering out instances where literal values could not be reconstructed through fuzzy match from the gold spans. After post-processing, SQUALL contained 11,276 table-question-answer triplets with logical form and lexical alignment annotations.

4 (State-of-the-Art)[4] Base Model: Seq2seq with Attention and Copying

Recent state-of-the-art text-to-SQL models extend the sequence-to-sequence (seq2seq) framework with attention and copying mechanisms (Zhong et al., 2017; Dong and Lapata, 2016, 2018; Suhr et al., 2020, *inter alia*). We adopt this strong neural paradigm as our base model. The seq2seq model generates one output token at a time via a probability distribution conditioned on both the input sequence representations and the partially-generated output sequence: $P(y \mid \mathbf{x}) = \prod_{i=1}^{|y|} P(y_i \mid \mathbf{y}_{<i}, \mathbf{x})$, where $\mathbf{x}$ and $\mathbf{y}$ are the feature representations for the input and output sequences, and $_{<i}$ denotes a prefix. The last token of y must be a special `<STOP>` token that terminates the output generation. The per-token probability distribution is modeled through Long-Short Term Memory networks (LSTMs, Hochreiter and Schmidhuber, 1997) and multi-layer perceptrons (MLPs):

$$\mathbf{h}_i = \text{LSTM}(\mathbf{h}_{i-1}, \mathbf{y}_{i-1}) \quad (1)$$

$$P(y_i \mid \mathbf{y}_{<i}, \mathbf{x}) = \text{softmax}\left(\text{MLP}(\mathbf{h}_i)\right). \quad (2)$$

The training objective is the negative log likelihood of the gold $y^\star$, defined for each timestep as

$$L_i^{\text{seq2seq}} = -\log P(y_i^\star \mid \mathbf{y}_{<i}^\star, \mathbf{x}).$$

Question and Table Encoding An input x contains a length-n question $q = q_1, \ldots, q_n$ and a table with m columns $c = c_1, \ldots, c_m$. The input question is represented through a bi-directional LSTM (bi-LSTM) encoder that summarizes information from both directions within the sequence. Inputs to the bi-LSTM are concatenations of word embeddings, character-level bi-LSTM vectors, part-of-speech embeddings, and named entity type embeddings. We denote the resulting feature vector associated with q_i as $\mathbf{q}_i$. For column names, the representation $\mathbf{c}_j$ concatenates the final hidden states of two LSTMs running in opposite directions that take the concatenated word embeddings, character encodings, and column data type embeddings as inputs. We also experiment with pre-trained BERT feature extractors (Devlin et al., 2019), where we feed the BERT model with the question and the columns as a single sequence delimited by the special `[SEP]` token, and we take the final-layer representations of the question words and the last token of each column as their representations.

Attention in Encoding To enhance feature interaction between the question and the table schema, for each question word representation $\mathbf{q}_i$, we use an attention mechanism to determine its relevant columns and calculate a linearly-weighted context vector $\widetilde{\mathbf{q}}_i$ as follows:

$$\widetilde{\mathbf{q}}_i = \text{Attn}(\mathbf{q}_i, \mathbf{c}) \triangleq \textstyle\sum_j \mathbf{a}_{ij}\mathbf{c}_j, \quad (3)$$

$$\text{where } \mathbf{a}_{ij} = \text{softmax}_j\left(\mathbf{q}_i^T W^{\text{att}} \mathbf{c}\right). \quad (4)$$

Then we run another bi-LSTM by concatenating the question representation $\mathbf{q}$ and context representation $\widetilde{\mathbf{q}}$ as inputs to derive a column-sensitive representation $\vec{\mathbf{q}}_i$ for each question word q_i. We apply a similar procedure to get the column representation $\vec{\mathbf{c}}_j$ for each column.

Attention in Decoding During decoding, to allow LSTMs to capture long-distance dependencies

[3] In the pilot study, the annotators independently labeled questions over the same 50 tables. We report the percentage of cases where one annotator accepted the other annotator's labels.

[4] In Appendix §B, we show that on SQUALL, our base model is competitive with a state-of-the-art system (Suhr et al., 2020) benchmarked on the Spider dataset (Yu et al., 2018).

from the input, we add attention-based features to the recurrent feature definition of Eq. (1):

$$\mathbf{v}_i = \text{Attn}(\mathbf{h}_i, \vec{\mathbf{q}}) \tag{5}$$

$$\mathbf{h}_i = \text{LSTM}(\mathbf{h}_{i-1}, [\mathbf{v}_{i-1}; \mathbf{y}_{i-1}]). \tag{6}$$

SQL Token Prediction with Copying Mechanism Since each output token can be an SQL keyword, a column name or a literal value, we factor the probability defined in Eq. (2) into two components: one that decides the type $t_i \in \{\mathsf{KEY}, \mathsf{COL}, \mathsf{STR}\}$ of y_i:

$$P(t_i \mid \mathbf{y}_{<i}, \mathbf{x}) = \text{softmax}\left(\text{MLP}^{\text{type}}(\mathbf{h}_i)\right),$$

and another that predicts the token conditioned on the type t_i. For token type KEY, we predict the keyword token with another MLP:

$$P(y_i \mid \mathbf{y}_{<i}, \mathbf{x}, t_i = \mathsf{KEY}) = \text{softmax}\left(\text{MLP}^{\mathsf{KEY}}(\mathbf{h}_i)\right).$$

For COL and STR tokens, the model selects directly from the input column names c or question q via a copying mechanism. We define a probability distribution with softmax-normalized bilinear scores:

$$P(y_i = c_j \mid \mathbf{y}_{<i}, \mathbf{x}, t_i = \mathsf{COL}) = \text{softmax}_j(\mathbf{s}_{i\cdot}),$$
$$\text{where } \mathbf{s}_{ij} = \mathbf{h}_i^\top W^{\mathsf{COL}} \mathbf{c}_j.$$

Similarly, we define literal string copying from q with another bilinear scoring matrix W^{STR}.

5 Using Alignments in Model Training

The model design in §4 includes many latent interactions within and across the encoder and the decoder. We now describe how our manual alignments can enable direct supervision on such previously latent interactions. Our alignments can be used as supervision for the necessary attention weights (§5.1). In an *oracle experiment* where we replace induced attention with manual alignments, the jump in logical form accuracy shows *alignments are valuable*, if only the models could reproduce them (§5.2). Moreover, alignments enable a column-prediction auxiliary task (§5.3).

The loss function L of our full model is a linear combination of the loss terms of the seq2seq model, supervised attention, and column prediction:

$$L = L^{\text{seq2seq}} + \lambda^{\text{att}} L^{\text{att}} + \lambda^{\text{CP}} L^{\text{CP}},$$

where we define L^{att} and L^{CP} below.

Attention type	ACC_{LF} (Dev)	Δ
Induced attention	37.8 ± 0.6	
Oracle attention		
Encoder only	51.5 ± 1.4	$+13.7$
Decoder only	49.4 ± 0.9	$+11.6$
Encoder + decoder	61.7 ± 0.4	$+23.9$

Table 2: Oracle experiment LF-accuracy results over five dev sets from random splits, where attention weights are replaced by manual alignments. *Induced attention* refers to the base model (§4).

5.1 Supervised Attention

Our annotated lexical alignments resemble our base model's attention mechanisms. At the encoding stage, question tokens and the relevant columns are aligned (e.g., "who" $\leftrightarrow$ column "athlete") which should induce higher weights in both question-to-column and column-to-question attention (Eq. (3) and Eq. (4)); similarly, for decoding, annotation reflects which question words are most relevant to the current output token. Inspired by improvements from supervised attention in machine translation (Liu et al., 2016; Mi et al., 2016), we train the base model's attention mechanisms to minimize the Euclidean distance[5] between the human-annotated alignment vector $\mathbf{a}^\star$ and the model-generated attention vector $\mathbf{a}$:

$$L^{\text{att}} = \frac{1}{2}\|\mathbf{a} - \mathbf{a}^\star\|^2.$$

The vector $\mathbf{a}^\star$ is a one-hot vector when the annotation aligns to a single element, or $\mathbf{a}^\star$ represents a uniform distribution over the subset in cases where the annotation aligns multiple elements.

5.2 Oracle Experiments with Manual Alignments

To present the potential of alignment annotations for models with supervised attention, we first assume a model that can flawlessly reproduce our annotations within the base model. During training and inference, we feed the true alignment vectors in place of the attention weights to the encoder and/or decoder. Table 2 shows the resultant logical form accuracies. Access to oracle alignments provides up to 23.9% absolute higher accuracy over the base model. This wide gap suggests the high potential for training models with our lexical alignments.

[5] See Appendix F for experiments with other distances.

Model	ACC_{EXE} (Test)
Prior work (all necessarily are weakly supervised)	
Single model	34.2–44.5
Single model (w/ BERT)	48.8
Ensemble	37.7–46.9
This paper (strongly supervised for the first time)	
Single model (ALIGN)	49.7 ± 0.4
Single model (ALIGN w/ BERT)	54.1 ± 0.2
Ensemble (ALIGN)	53.1
Ensemble (ALIGN w/ BERT)	57.2

Table 3: WTQ test set execution accuracies (%). The accuracy ranges for prior work are aggregated over Pasupat and Liang (2015), Neelakantan et al. (2016), Krishnamurthy et al. (2017), Zhang et al. (2017), Haug et al. (2018), Liang et al. (2018), Dasigi et al. (2019), Agarwal et al. (2019), Wang et al. (2019), and Herzig et al. (2020). Unsurprisingly, our models trained on SQUALL surpass weakly-supervised previous work.

5.3 Column Prediction

Wang et al. (2019) show the importance of inferring token-column correspondence in a weakly-supervised setting; SQUALL enables full supervision for an auxiliary task that directly predicts the corresponding column c_j for each question token q_i. We model this auxiliary prediction as:

$$\mathbf{s}_{ij} = \mathbf{q}_i^\top W^{\text{CP}} \mathbf{c}_j$$
$$P(q_i \text{ matches } c_j \mid q_i) = \text{softmax}_j(\mathbf{s}_{i \cdot}).$$

For the corresponding loss L^{CP} over tokens that match columns, we use cross-entropy.

Exact-match Features: An Unsupervised Alternative A heuristic-based, albeit lower-coverage, alternative to manual alignment is to use questions' mentions of column names. Thus, we use automatically-generated exact-match features in our baseline models for comparison in our experiments. For question encoders, we include two embeddings derived from binary exact-match features: indicators of whether the token appears in (1) any of the column headers and (2) any of the table cells. Similarly, for the column encoders, we also include an exact-match feature of whether the column name appears in the question.

6 Experiments

Setup We randomly shuffle the tables in SQUALL and divide them into five splits. For each setting, we report the average logical form accuracy ACC_{LF} (output LF exactly matches the target LF) and execution accuracy ACC_{EXE} (output LF may not match the target LF, but its execution yields the gold-standard answer) as well as the standard deviation of five models, each trained with four of the splits as its training set and the other split as its dev set. We denote the base model from §4 as SEQ2SEQ and our model trained with both proposed training strategies in §5 as ALIGN. The main baseline model we compare with, SEQ2SEQ^{+}, is the base model enhanced with the automatically-derived exact-match features (§5.3). See Appendix Appendix A for model implementation details.

Model	Dev ACC_{LF}	Dev ACC_{EXE}	Test ACC_{EXE}
SEQ2SEQ^{+}	37.8 ± 0.6	56.9 ± 0.7	46.6 ± 0.5
ALIGN	42.2 ± 1.5	61.3 ± 0.8	49.7 ± 0.4
SEQ2SEQ^{+} w/ BERT	44.7 ± 2.1	63.8 ± 1.1	51.8 ± 0.4
ALIGN w/ BERT	47.2 ± 1.2	66.5 ± 1.2	54.1 ± 0.2

Table 4: Logical form (ACC_{LF}) and execution (ACC_{EXE}) accuracies (%) on dev and test sets, showing the utility of learning from lexical supervisions.

WTQ Test Results Table 3 presents the WTQ test-set ACC_{EXE} of ALIGN compared with previous models. Unsurprisingly, SQUALL's supervision allows our models to surpass weakly supervised models. Single models trained with BERT feature extractors exceed prior state-of-the-art by 5.3%. However, our main scientific interest is not these numbers per se, but how beneficial additional lexical supervision is.

Effect of Alignment Annotations To examine the utility of lexical alignments as a finer-grained type of supervision, we compare ALIGN with SEQ2SEQ^{+} in Table 4. Both have access to logical form supervision, but ALIGN additionally uses lexical alignments during training. ALIGN improves SEQ2SEQ by 2.3% with BERT and 3.1% without, showing that lexical alignment annotation is more beneficial than automatically-derived exact-match column reference features.[6]

Effect of Individual Strategies Table 5 compares model variations. We add each individual training strategy into the baseline SEQ2SEQ^{+} model and ablate components from the ALIGN model. Each component contributes to increased accuracies compared with SEQ2SEQ^{+}. The effects range from +1.3% ACC_{EXE} with column prediction to

[6] Test set accuracies are lower than on the dev set because the WTQ test set includes questions unanswerable by SQL.

Component	Dev ACC$_{LF}$	Dev ACC$_{EXE}$
SEQ2SEQ	31.0 ± 0.7	48.8 ± 0.8
SEQ2SEQ^{+}	37.8 ± 0.6	56.9 ± 0.7
+ Supervised decoder attn.	39.4 ± 1.1	58.6 ± 1.3
+ Supervised encoder attn.	41.3 ± 1.7	60.7 ± 0.7
+ Column prediction	38.6 ± 0.5	58.2 ± 0.8
ALIGN	42.2 ± 1.5	61.3 ± 0.8
- Supervised decoder attn.	41.6 ± 1.8	61.1 ± 1.3
- Supervised encoder attn.	39.6 ± 0.6	58.7 ± 0.8
- Column prediction	41.8 ± 1.6	60.9 ± 0.8
- Exact-match features	39.5 ± 1.1	58.8 ± 0.7
Oracle attention	61.7 ± 0.4	–

Table 5: Dev logical form (ACC$_{LF}$) and execution (ACC$_{EXE}$) accuracies for different model variations (w/o BERT). The superimposed bar chart provides a visual presentation of ACC$_{LF}$. Each ALIGN component contributes to increased accuracies compared with SEQ2SEQ^{+}, while the oracle attention model demonstrates the unrealized potential of the alignments.

	ACC$_{LF}$	ACC$_{TEMP}$	ACC$_{COL}$
SEQ2SEQ^{+}	37.8	64.7	39.6
ALIGN	42.2	66.7	44.5
(delta)	(+4.4)	(+2.0)	(+4.9)

Table 6: Dev logical form (ACC$_{LF}$), template (ACC$_{TEMP}$) and column (ACC$_{COL}$) accuracies. Parenthetical numbers are deltas with respect to the baseline. ALIGN improves ACC$_{COL}$ the most.

Unseen Templates	ACC$_{LF}$	ACC$_{EXE}$
SEQ2SEQ^{+}	15.5	44.8
ALIGN	26.1	57.3

Table 7: Model accuracies in a generalization setting: we exclude an SQL template from training, and evaluate on that unseen template. Shown are macro-averages over the 10 most frequent templates. ALIGN is more accurate than SEQ2SEQ^{+} by a large margin.

+3.8% ACC$_{EXE}$ with supervised encoder attention. Supervised encoder attention is the single most effective strategy: including it produces the highest gains and ablating it the largest drop. The exact-match column reference features are essential to the baseline model: SEQ2SEQ without those features has 8.1% lower ACC$_{EXE}$. Nonetheless, supervised encoder attention and column prediction are still effective on top of the exact-match features. Yet, ALIGN's accuracy is still far below that of the oracle models; we hope SQUALL can inspire future work to take better advantage of its rich supervision.

Effect of Annotation Availability: Are Lexical Alignments Worth It? The lefthand side of Figure 2 plots SEQ2SEQ^{+}'s and ALIGN's learning curves. For each of for SEQ2SEQ^{+}'s accuracy levels, ALIGN reaches a similar level but at the much "cheaper" training cost of about half as many training examples. Moreover, the righthand side of Figure 2 shows what happens if ALIGN has access to all the training logical forms, but only a percentage of the accompanying alignments. Surprisingly, more than half of the accuracy improvement comes from as little as 5% of the alignment annotations. Because the cost of aligning an example is less than half of that for writing a logical form (§3.1), we conclude that annotating lexical alignments is a cost-effective approach on a fixed budget.

Where Do Our Models Improve the Most? According to Table 6, ALIGN produces the highest gains with respect to SEQ2SEQ^{+} on the subtask of column selection (+4.9%), compared with a +2.0% improvement on generating correct SQL templates. The gain is larger on complex SQL templates (i.e., those with more aggregation functions and nested queries).[7] which demonstrates the effectiveness of reinforcing question-column correspondence through supervised attention and a column prediction auxiliary task.

Do Our Models Generalize Better to Unseen Query Templates? We follow Finegan-Dollak et al. (2018) and consider a challenging evaluation setting where the models are tested on unseen SQL query templates. In Table 7, ALIGN shows an even larger margin compared with SEQ2SEQ^{+} in this setting, suggesting that lexical alignment supervision benefits model robustness. See Appendix I for detailed results.

Are the Induced Attention Weights Similar to Manual Alignments? Table 8 quantitatively compares the attention distributions. The models trained with and without supervised attention have very different attention patterns: without explicit supervision, the models focus on a few items (low entropy values), but those items are usually unlike manually-derived alignments (low recall). Interestingly, the supervised decoder attention encourages the model to induce question-to-column (q2c) attention that seems similar to human alignment

[7] For example, on template `SELECT COUNT(col) FROM w`, the ACC$_{COL}$ is 59.4 (ALIGN) vs. 48.9 (SEQ2SEQ^{+}). See Appendix §H for detailed result breakdowns.

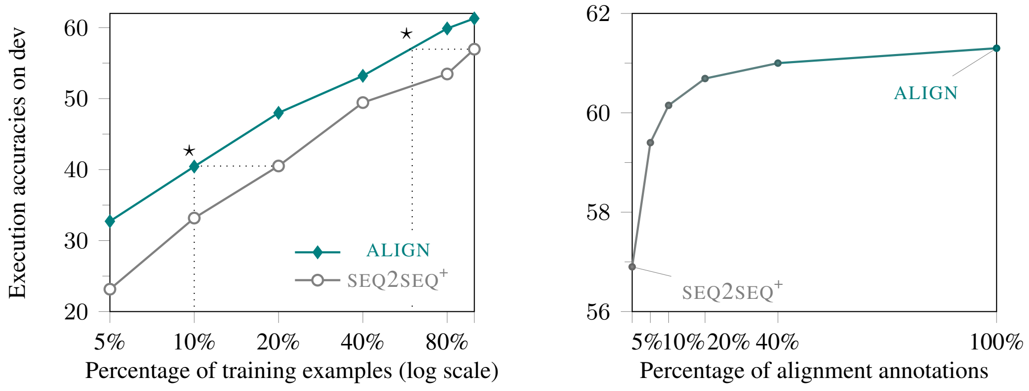

Figure 2: (Left) the ⋆ markers on the learning curves illustrate that ALIGN uses roughly half the amount of training data to achieve similar ACC_{EXE} as SEQ2SEQ$^+$. (Right) annotating just 5% of the logical forms with alignments yields *half* of the accuracy improvement of ALIGN.

Model	Recall			Entropy		
	q2c	c2q	d2q	q2c	c2q	d2q
SEQ2SEQ$^+$	26.1	4.8	33.2	0.31	0.16	1.24
+ Sup. enc.	64.8	66.0	35.6	1.57	1.95	1.10
+ Sup. dec.	55.5	3.9	86.6	0.44	0.24	0.99
ALIGN	65.4	65.9	86.2	1.56	1.94	1.00

Table 8: Recall against hand-annotated alignments and average entropy of the attention distributions in the question-to-column (q2c), column-to-question (c2q) and decoder-to-question (d2q) modules, comparing models trained with supervised encoder/decoder attention, none (SEQ2SEQ$^+$), or both strategies (ALIGN).

Serial Name (c1)	*Role (c2)*	*Co-Star (c3)*	*Channel (c4)*	...
Saat Phere	Nahar Singh	Rajshree Thakur	Zee TV	...
Nach Baliye 2	Himself	Keerti Gaekwad Kelkar	Star One	...

Question:

What was the only ①show that ran on the ②channel ③star one?

Target:

```
SELECT ①c1 FROM w where ②c4 = ③'star one'
```

SEQ2SEQ$^+$:

```
SELECT ?c3 FROM w where ②c4 = ③'star one'
```

ALIGN:

```
SELECT ①c1 FROM w where ②c4 = ③'star one'
```

Figure 3: An example with SEQ2SEQ$^+$ and ALIGN predictions. SEQ2SEQ$^+$ selects an incorrect column.

judgments. This is an arguably surprising benefit, since the supervised decoder was not trained with q2c supervision, and so one might have expected it to perform similarly to SEQ2SEQ$^+$. However, one needs to be careful in interpreting these results, as machine-induced attention distributions are not intended for direct human interpretation (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019).

Qualitative Analysis Our additional supervision helps when the question has little textual overlap with the referred columns. Figure 3 shows an example. With finer-grained supervision, ALIGN learns the column "Serial Name" corresponds to the question word "show", but SEQ2SEQ$^+$ selects the wrong column "Co-Star".

7 Related Work

Attention and Alignments Explicit supervision for attention mechanisms (Bahdanau et al., 2015) is helpful for many tasks, including machine translation (Liu et al., 2016; Mi et al., 2016), image captioning (Liu et al., 2017), and visual question answering (Gan et al., 2017). For semantic parsing, Rabinovich et al. (2017) improve code generation with exact string-match heuristics to provide supervision for attention. Wang et al. (2019) argue that structured alignment is crucial to text-to-SQL models and they induce latent alignments in a weakly-supervised setting. In contrast, we take a fully-supervised approach and train models with manual alignments.

Lexical Focus and Semantic Parsing Our lexical alignment annotations are similar to semantic lexicons in lexicalized-grammar-based semantic parsing (Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010; Krishnamurthy and Mitchell, 2012; Artzi and Zettlemoyer, 2013). Those lexicons are usually well-typed to support semantic composition. It is an interesting future direction to explore how to model analogous compositional aspects with our type-flexible alignments through, for example, syntax-based alignment (Zhang and Gildea, 2004).

Annotator Rationales A related direction to enriching annotations is supplying annotator rationales (Zaidan et al., 2007), i.e., evidence supporting the annotations in addition to the final labels. Many recent datasets on machine reading comprehension and question answering, such as HotpotQA (Yang et al., 2018) and CoQA (Reddy et al., 2019), include such intermediate annotations at dataset release. Dua et al. (2020) show that these annotator rationales improve model accuracy for a given annotation budget on machine reading comprehension. The alignments we provide could, at a stretch, be considered a type of rationale for the output SQL annotation.

Text-to-SQL Datasets There is growing interest in both the database and NLP communities in text-to-SQL applications. Widely-used domain-specific datasets include ATIS (Price, 1990; Dahl et al., 1994), GeoQuery (Zelle and Mooney, 1996; Popescu et al., 2003), Restaurants (Tang and Mooney, 2000; Popescu et al., 2003), and Scholar (Iyer et al., 2017). WikiSQL (Zhong et al., 2017) is among the first large-scale datasets with question-logical form pairs querying a wide range of data tables extracted from Wikipedia, but WikiSQL's logical forms are generated from a limited set of templates. In contrast, WTQ questions are authored by humans under no specific constraints, and as a result WTQ includes more diverse semantics and logical operations. The family of Spider datasets (Yu et al., 2018, 2019a,b) contain queries even more complex than in WTQ, including a higher percentage of nested queries and multiple table joins. We leave extensions of lexical alignments to Spider's complex-structure queries to future work.

8 Conclusion

We introduce SQUALL, the first large-scale semantic parsing dataset with both hand-produced target logical forms and manually-derived lexical alignments between questions and SQL queries. Our dataset enables finer-grained supervision than existing datasets have previously supported. We incorporate the alignments into encoder-decoder-based neural models through supervised attention and an auxiliary task of column prediction. Experiments confirm our intuition that finer-grained supervision is helpful to model training. Our oracle studies also show that there is large unrealized further potential for our annotations. Thus, it remains an exciting challenge for future research to use our lexical alignment annotations more effectively.

Our annotation cost analysis shows that collecting additional lexical alignments is more cost-effective for improving model accuracy than having only logical forms. We hope that our findings will help future dataset design decisions and extensions of other existing datasets. One potential future direction is to further investigate the utility of lexical alignments in a cross-dataset/domain evaluation setting (Suhr et al., 2020).

Acknowledgments

We thank the members of UMD CLIP, Xilun Chen, Jack Hessel, Thomas Müller, Ana Smith, and the anonymous reviewers and meta-reviewer for their suggestions and comments. TS was supported by a Bloomberg Data Science Ph.D. Fellowship. CZ and JBG are supported by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL), and awarded to Raytheon BBN Technologies under contract number FA865018-C-7885. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

References

Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. 2019. Learning to generalize from sparse and underspecified rewards. In *Proceedings of the International Conference of Machine Learning*, pages 130–140.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, pages 49–62.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, pages 43–48.

Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke Zettlemoyer, and Eduard Hovy. 2019. Iterative search for weakly supervised semantic parsing. In *Conference of the North American Chapter of the*

Association for Computational Linguistics, pages 2669–2680.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the Association for Computational Linguistics*, pages 33–43.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the Association for Computational Linguistics*, pages 731–742.

Dheeru Dua, Sameer Singh, and Matt Gardner. 2020. Benefits of intermediate annotations in reading comprehension. In *Proceedings of the Association for Computational Linguistics*, pages 5627–5634.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the Association for Computational Linguistics*, pages 351–360.

Chuang Gan, Yandong Li, Haoxiang Li, Chen Sun, and Boqing Gong. 2017. VQS: Linking segmentations to questions and answers for supervised attention in VQA and question-focused semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1811–1820.

Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. 2018. Neural multi-step reasoning for question answering on semi-structured tables. In *European Conference on Information Retrieval*, pages 611–617.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the Association for Computational Linguistics*, pages 4320–4333.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the Association for Computational Linguistics*, pages 963–973.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 3543–3556.

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

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1516–1526.

Jayant Krishnamurthy and Tom M. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 754–765.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1223–1233.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *Proceedings of Advances in Neural Information Processing Systems*, pages 10015–10027.

Chenxi Liu, Junhua Mao, Fei Sha, and Alan Yuille. 2017. Attention correctness in neural image captioning. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, pages 4176–4182.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of International Conference on Computational Linguistics*, pages 3093–3102.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 2283–2288.

Arvind Neelakantan, Quoc V. Le, Martin Abadi, Andrew McCallum, and Dario Amodei. 2016. Learning a natural language interface with Neural Programmer. In *Proceedings of the International Conference on Learning Representations*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the Association for Computational Linguistics*, pages 1470–1480.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. In *Proceedings of the Association for Computational Linguistics*, pages 23–32.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *International Conference on Intelligent User Interfaces*, pages 149–157.

P. J. Price. 1990. Evaluation of spoken language systems: The ATIS domain. In *Proceedings of the Workshop on Speech and Natural Language*, pages 91–95.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the Association for Computational Linguistics*, pages 1139–1149.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

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

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 133–141.

Bailin Wang, Ivan Titov, and Mirella Lapata. 2019. Learning semantic parsers from denotations with latent structured alignments and abstract programs. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 3765–3776.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the Association for Computational Linguistics*, pages 1332–1342.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 11–20.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the Association for Computational Linguistics*, pages 1341–1350.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 2369–2380.

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. 2019a. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1962–1979.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 3911–3921.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the Association for Computational Linguistics*, pages 4511–4523.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 260–267.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, pages 1050–1055.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 658–666.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 678–687.

Hao Zhang and Daniel Gildea. 2004. Syntax-based alignment: Supervised or unsupervised? In *Proceedings of International Conference on Computational Linguistics*, pages 418–424.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. AMR parsing as sequence-to-graph transduction. In *Proceedings of the Association for Computational Linguistics*, pages 80–94.

Yuchen Zhang, Panupong Pasupat, and Percy Liang. 2017. Macro grammars and holistic triggering for efficient semantic parsing. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1214–1223.

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

A Model Implementation Details

We use and compare two different feature extractors in our experiments. For bi-LSTM encoders, we concatenate 100-dimensional word embeddings initialized from pre-trained GLoVE embeddings (Pennington et al., 2014), 8-dimensional part-of-speech and 8-dimensional named-entity embeddings as input to the LSTM encoders. Tokens that appear less than five times are replaced with a special "UNK" token. For the BERT setting, we fine-tune a $\text{BERT}_{\text{base}}$ model[8] and use the 768-dimensional final-layer representations. For the decoder, we embed previously decoded tokens, such as keywords, into 256-dimensional vectors and feed them as next-timestep input to the decoder LSTM. Both the encoder and decoder LSTMs have 128 hidden units and 2 layers. If the decoder predicts question words as literal strings in the output SQL queries, we replace them with the most similar table cell values using fuzzy match.[9] We set both λ^{att} and λ^{CP} to be 0.2. During training, we use a batch size of 8 and we set the dropout rate to be 0.3 in all MLPs and LSTMs. We use the Adam optimizer (Kingma and Ba, 2015) with default learning rate 0.001 and we clip gradients to 5.0. We train our models for up to 50 epochs and conduct early stopping based on per-epoch dev-set evaluation. On a single GTX 1080 Ti GPU, a training mini-batch takes 0.7 second on average and the training process finishes within 10 hours. We do not tune hyper-parameters.

B Comparison of Our Baseline Model with a State-of-the-Art Text-to-SQL Parser

To evaluate the strength of our baseline model, we compare it with Suhr et al.'s (2020) state-of-the-art model previously tested on the Spider dataset (Yu et al., 2018). Our task formulation is unlike the Spider dataset in that 1) the official Spider evaluation does not require predictions of literal values and 2) on our dataset, the model needs to predict data types for each column (e.g., _number in Figure 1). Suhr et al.'s (2020) model has already addressed the first difference by including literal string prediction modules, and we loosen our evaluation criteria for the sake of this comparison. We train Suhr et al.'s (2020) model on SQUALL with their reported hyperparameters and evaluate with a variant of logical form accuracies ($\text{ACC}^{-}_{\text{LF}}$) that accepts column type disparities between the prediction and the gold standard; Table B1 shows the evaluation results. Our baseline SEQ2SEQ^{+} model has competitive $\text{ACC}^{-}_{\text{LF}}$ with Suhr et al.'s (2020) state-of-the-art text-to-SQL parser.

Model	$\text{ACC}^{-}_{\text{LF}}$
SEQ2SEQ^{+} w/ BERT	50.8
Suhr et al. (2020) w/ BERT	51.7

Table B1: Dev logical form accuracy excluding column type ($\text{ACC}^{-}_{\text{LF}}$) of our SEQ2SEQ^{+} w/ BERT is comparable to that of a state-of-the-art model on Spider.

C Annotation Guidelines

In our pilot study, we instruct two expert SQL annotators to write down SQL equivalents of the English questions and to pick out the lexical mappings between the question and SQL tokens that correspond to each other semantically and are atomic, i.e., they cannot further decompose into smaller meaningful mappings. These underspecified instructions lead to 70.4% agreement on SQL annotation and 75.1% agreement on alignment annotation. The annotators have similar but not identical intuitions about, for example, what constitutes an atomic unit, especially when there are equally plausible alternative options. Following discussions, we refine our annotation guidelines for frequently occurring patterns to ensure consistent annotations, as follows:

General Rules

1. SQL queries should reflect the semantic intent of the English questions, even if shorter SQL queries return the same execution results. The only exception is when SQL offers no straightforward implementation of the implicit semantic constraints. In that case, answer the first appearing subquestion, i.e., assume that the implicit semantic constraints are always met. For example, it is implicitly assumed in the question "which city are A and B located in?" that A and B are located in the same city; write down the SQL equivalent for "which city is A located in?".
2. When there are competing choices of annotation, select the simplest version. Among alternative SQL queries, select the one with fewer nestings and fewer SQL tokens: `SELECT MAX(col) FROM w` is prioritized over `SELECT`

[8] https://github.com/huggingface/transformers
[9] https://github.com/seatgeek/fuzzywuzzy

col FROM w ORDER BY col DESC LIMIT 1. Following this rule, default values are always omitted since the queries are shorter without them. These include, for example, the keyword ASC in an ORDER BY clause.

3. Lexical alignments should cover as many semantically-meaningful tokens as possible, even if there is no word overlap. For example, for the question "who performed better, toshida or young-sun?", align the word "performed" to its corresponding column ("result" or "rank"). For *wh*-tokens, align "when", "who" and "where" if appropriate, but omit alignments of "what" and "which" when they do not contribute to concrete meanings.
4. Prioritize alignments with exact lexical matches. This means that for many noun phrases, align bare nouns excluding the determiners instead of maximal noun phrases (e.g., "movie" rather than "the movie" should be aligned to the "movie" column token in the SQL query). In contrast, include "the" in the alignment of superlatives (e.g., "the least"), since superlatives usually do not lexically overlap with the column tokens.
5. In general, the annotation should not depend on the table contents and sorting assumptions. In other words, use direct references to the presented row order id as little as possible. However, use id if the question explicitly asks about the presentation order, e.g., "the first on the list" or "the first listed".

Some Frequent Specific Cases

1. Align "how many" to the aggregation operation when appropriate, but do not align "how many" when the SQL query directly selects a column without aggregation, e.g., the question is "how many total medals has Spain won?" and the table contains a column "total".
2. Only add the keyword DISTINCT if there are clear linguistic cues ("how many *different* countries on the table?"), otherwise do not use DISTINCT.
3. Use COUNT(col) if possible and use COUNT(*) only if there is no good match from the question to any column.
4. When the question asks about the row with the max/min value in a column, generally use SELECT col FROM w ORDER BY col [DESC] LIMIT 1. If there are ties in the max/min values, use SELECT col FROM w WHERE col = (SELECT MAX(col) FROM w).
5. Align question word "game" to "date" column if necessary but use COUNT(*) for counting the game numbers when there are no better alignment alternatives.
6. Align words referring to performance, such as "fast", to the corresponding "result"/"time" columns; if not available, align them to "rank" columns that indirectly refer to performance; if still not available, align them to id, which explicitly relies on the table being presorted by the performance.

D Database Construction

We assume 9 basic data types for WTQ tables: numbers (e.g., "5"), numbers with units (e.g., "5 kg") , date and time (e.g., "May 29, 1968", "3:56"), (sports) scores (e.g., "w 5:3"), number spans (e.g., "12–89"), time spans (e.g., "May 2011–June 2012"), fractions (e.g., "3/5") , street addresses (e.g., "2020 Westchester Street"), and raw texts (e.g., "John Shermer"). Additionally, we consider two composite types: binary tuples (e.g., "KO (head kick)") and lists (e.g., "Wojtek Fibak, Joakim Nyström"). Binary tuples are split into two subcolumns in the generated databases, and lists are automatically transformed to a separate table joined with the original table through primary-foreign key relations. Data types for each column are first identified with regular expressions and manually verified by annotators. Any column that contains a type outside of these 9 types is interpreted as raw text. We also filter out aggregation rows from the tables so that the SQL aggregation functions over the table can skip those pre-computed aggregates.

E Additional Alignment Data Statistics

Table E2 shows the part-of-speech tags that are most- and least-aligned.[10] Comparative and superlative adjectives and adverbs are among the most frequently aligned tokens, while pronouns and function words are infrequently aligned.

[10] These POS tags are automatically derived from Stanford CoreNLP toolkit and are provided in the WTQ dataset.

POS	(%)↓	POS	(%)↑
RBS (Adverb, superlative)	99.02	. (Punctuation)	0.15
JJR (Adjective, comparative)	96.24	WDT (*wh*-determiner)	1.20
JJS (Adjective, superlative)	94.66	VBD-AUX (Auxiliary verb)	2.26
RBR (Adverb, comparative)	93.89	EX (Existential *there*)	3.56
WRB (*wh*-adverb)	88.25	PRP$ (Possessive pronoun)	9.38
JJ (Adjective)	82.07	POS (Possessive ending)	13.42
CD (Cardinal number)	79.48	PRP (Personal pronoun)	13.95
NNP (Proper noun, singular)	75.70	WP (*wh*-pronoun)	20.58

Table E2: The POS tags with the highest and lowest alignment ratios (%) to SQL queries (with more than 100 occurrences). Comparative/superlative adjectives (JJR, JJS) and adverbs (RBS, RBR) are most aligned, corresponding to SQL operations like MAX. Punctuations (.), *wh*-determiners (WDT), helper-verbs (VBD-AUX), existential *there*'s (EX), and pronouns (PRP, PRP$) are least aligned.

Attention Loss	ACC_{LF}	ACC_{EXE}
Mean squared error (ALIGN)	41.8 ± 1.6	60.9 ± 0.8
Multiplication	40.3 ± 1.5	59.4 ± 1.0
Cross entropy	41.6 ± 1.2	60.3 ± 1.0

Table F3: Dev logical form (ACC_{LF}) and execution (ACC_{EXE}) accuracies with different attention loss functions. Our final model ALIGN uses mean squared error, the most accurate variant of the three loss functions.

Model	Dev	
	ACC_{LF}	ACC_{EXE}
SEQ2SEQ$^+$	37.8 ± 0.6	56.9 ± 0.7
ALIGN (Heuristics)	40.3 ± 1.8	59.6 ± 1.4
ALIGN (Manual)	42.2 ± 1.5	61.3 ± 0.8

Table G4: Dev logical form (ACC_{LF}) and execution (ACC_{EXE}) accuracies comparing ALIGN trained with automatic and manual alignments. Training with automatic alignments leads to higher accuracies than SEQ2SEQ$^+$ and manual annotations give an additional accuracy improvement.

F Different Loss Functions for Supervised Attention

Following Liu et al. (2016), we experiment with three different attention loss definitions:

$$L^{\text{att}} = \frac{1}{2}\|\mathbf{a} - \mathbf{a}^\star\|^2 \quad \text{(Mean Squared Error)}$$
$$L^{\text{att}} = -\log(\mathbf{a} \cdot \mathbf{a}^\star) \quad \text{(Multiplication)}$$
$$L^{\text{att}} = -\mathbf{a}^\star \cdot \log(\mathbf{a}), \quad \text{(Cross Entropy)}$$

where $\mathbf{a}_i$ and $\mathbf{a}_i^\star$ denote the learned attention weights and annotated gold-standard alignments. A smaller distance between $\mathbf{a}_i$ and $\mathbf{a}_i^\star$ indicates a model better at reproducing our alignment annotation. While both mean squared error and multiplication are symmetric in $\mathbf{a}_i$ and $\mathbf{a}_i^*$, cross entropy is asymmetric and has been previously shown to be the most effective measure in the task of machine translation (Liu et al., 2016). Table F3 shows dev-set results with different supervised attention loss choices in ALIGN's encoder. The mean square error loss is the strongest, with 1.5% higher execution accuracy than multiplication loss and 0.6% higher than cross-entropy loss.

G ALIGN Trained with Heuristically-Generated Alignments

We experiment with question-column alignments derived from textual fuzzy matching between column names and question 5-grams. Table G4 shows dev-set results. Training with automatic alignments improves over the SEQ2SEQ$^+$ model, but manual annotations provide an additional $+1.7\%$ ACC_{EXE}. The manual annotations are cleaner and more informative since there are many column mentions without any lexical overlap with the column headers (e.g., "who" ↔ column "athlete").

H Template-based Evaluation

Table H5 shows dev-set results of the top 10 most frequent templates. We report logical form (ACC_{LF}), template (ACC_{TEMP}) and column (ACC_{COL}) accuracies. ACC_{COL} is calculated on the subset where template predictions are accurate.[11] The improvement of ALIGN over SEQ2SEQ$^+$ is more significant on ACC_{COL} than ACC_{TEMP}. Additionally, ALIGN tends to yield higher ACC_{COL} gains on complex templates, compared with simple and common templates.

[11] We do not include literal string and number accuracies: both SEQ2SEQ$^+$ and ALIGN get nearly perfect scores ($> 98\%$).

Template	Count	ACC$_{\text{LF}}$ SEQ2SEQ^{+}	ACC$_{\text{LF}}$ ALIGN	ACC$_{\text{TEMP}}$ SEQ2SEQ^{+}	ACC$_{\text{TEMP}}$ ALIGN	ACC$_{\text{COL}}$ SEQ2SEQ^{+}	ACC$_{\text{COL}}$ ALIGN
`SELECT col FROM w ORDER BY col [DESC] LIMIT 1`	1,490	48.1	**50.6**	86.9	**87.6**	56.3	**60.2**
`SELECT col FROM w WHERE col = STR`	1,149	39.5	**42.6**	73.6	**75.0**	40.1	**44.0**
`SELECT COUNT(col) FROM w WHERE col = STR`	1,127	55.0	**59.8**	85.2	**86.1**	55.9	**60.3**
`SELECT COUNT(col) FROM w WHERE col COMP NUM`	635	50.1	**57.6**	89.0	**91.1**	57.8	**66.0**
`SELECT col FROM w WHERE col = NUM`	607	49.4	**54.7**	72.9	**75.3**	49.7	**55.0**
`SELECT COUNT(col) FROM w`	507	43.2	**51.3**	**78.1**	77.7	48.9	**59.4**
`SELECT col FROM w GROUP BY col ORDER BY COUNT(col) [DESC] LIMIT 1`	315	34.6	**47.3**	80.0	**85.4**	36.2	**49.5**
`SELECT COUNT(col) FROM w WHERE col = NUM`	308	51.0	**59.8**	85.1	**87.3**	51.9	**59.7**
`SELECT col FROM w WHERE col = (SELECT col FROM w WHERE col = STR) + 1`	284	61.2	**61.6**	**76.1**	75.7	61.6	**62.0**
`SELECT col FROM w WHERE col IN (STR, STR) ORDER BY col [DESC] LIMIT 1`	282	39.0	**46.8**	85.5	**85.8**	49.3	**56.0**
Entire Corpus	11,276	37.8	**42.2**	64.7	**66.7**	39.6	**44.5**

Table H5: Dev logical form (ACC$_{\text{LF}}$), template (ACC$_{\text{TEMP}}$) and column (ACC$_{\text{COL}}$) accuracies on the 10 most frequent templates. We combine model predictions from five data splits for this analysis. `[DESC]` denotes the keyword `DESC` is optional, and `COMP` includes comparison operators ($>$, $<$, $>=$, $<=$ and $\neq$). ALIGN yields higher ACC$_{\text{COL}}$ gains on complex templates, compared with simple and common templates.

Unseen Template	Count	ACC$_{\text{LF}}$ SEQ2SEQ^{+}	ACC$_{\text{LF}}$ ALIGN	ACC$_{\text{EXE}}$ SEQ2SEQ^{+}	ACC$_{\text{EXE}}$ ALIGN
`SELECT col FROM w ORDER BY col [DESC] LIMIT 1`	1,490	9.0	**23.1**	38.9	**48.2**
`SELECT col FROM w WHERE col = STR`	1,149	**12.8**	11.3	48.8	**53.7**
`SELECT COUNT(col) FROM w WHERE col = STR`	1,127	9.0	**34.0**	32.0	**57.0**
`SELECT COUNT(col) FROM w WHERE col COMP NUM`	635	22.6	**45.2**	51.6	**58.9**
`SELECT col FROM w WHERE col = NUM`	607	15.4	**19.5**	58.5	**68.3**
`SELECT COUNT(col) FROM w`	507	0.0	**1.0**	19.0	**23.0**
`SELECT col FROM w GROUP BY col ORDER BY COUNT(col) [DESC] LIMIT 1`	315	3.3	**50.8**	24.6	**73.8**
`SELECT COUNT(col) FROM w WHERE col = NUM`	308	**34.0**	30.0	59.0	**66.0**
`SELECT col FROM w WHERE col = (SELECT col FROM w WHERE col = STR) + 1`	284	**30.8**	15.4	**61.5**	57.7
`SELECT col FROM w WHERE col IN (STR, STR) ORDER BY col [DESC] LIMIT 1`	282	17.9	**30.4**	53.6	**66.4**
Macro-average over the above templates	—	15.5	**26.1**	44.8	**57.3**

Table I6: Dev logical form (ACC$_{\text{LF}}$) and execution (ACC$_{\text{EXE}}$) accuracies in a generalization evaluation setting following Finegan-Dollak et al. (2018), where instances of a given template are ablated from training, and we evaluate model accuracies on that unseen template. ALIGN outperforms SEQ2SEQ^{+} in ACC$_{\text{EXE}}$ on 9 out of the 10 most frequent templates.

I Evaluation Results on Unseen SQL Templates

Table I6 considers an evaluation setting of Finegan-Dollak et al. (2018) to test the model accuracies on unseen SQL templates. We exclude all instances of a given template from the training set, and then evaluate only on that template. ALIGN outperforms SEQ2SEQ^{+} in ACC$_{\text{EXE}}$ on 9 out of the 10 most frequent templates. Notably, on a template that contains both `GROUP BY` and `ORDER BY` clauses, the ACC$_{\text{EXE}}$ improvement of ALIGN is as large as +49.2%.

References

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the Association for Computational Linguistics*, pages 351–360.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of International Conference on Computational Linguistics*, pages 3093–3102.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1532–1543.

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

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 3911–3921.