Model concept

TTS: Text-to-SQL
RTR: Retrieve-then-Reason/End-to-End
FT: Fine-tuning
ICL: In-Context Learning

Paper	Pre train	En coding	De coding	Prompt ing	Test	Design
TaBERT (2020)	✅	✅			TTS-FT, RTR-FT	• Pretrained by masked language reconstruction on 26M examples (English Wikipedia & WDC WebTable); • Encode table content snapshot only, top K rows most relevant to text (highest n-gram overlap); • Propose vertical self-attention for info flow across cell representations of different rows;
TAPAS (2020)	✅	✅	✅		RTR-FT	• Pretrain by masked language reconstruction (whole word and whole cell) on 3.3M Infobox and 2.9M WikiTable tables; • Additional positional embeddings for tabular structure; • Two classification layers for selecting cells and aggregation operators that operate on the cells; • Surrounding texts of the tables as a proxy of natural language utterance;
RAT-SQL (2020)		✅	✅		TTS-FT	• Encode schema as a directed graph (col and tbl as nodes, relation as edge) and question word; • Link schema and question based on name (match type) and value; • Decode a SQL abstract syntax tree in depth-first traversal order then use LSTM;
LGESQL (2021)		✅	✅		TTS-FT	• Propose dual relational graph attention layer combining node-centric graph and line graph, where node embeddings in one graph play the role of edge features in another graph; • Add one decoder for auxiliary “graph pruning” task (distinguish irrelevant schema items from golden schema items used in the target query);
GAP (2021)	✅				TTS-FT	• Pretrain by column prediction (whether used in the input utterance), column recovery (replace col names with cell value, recover from utterance or cell value), SQL generation given utterances and schema; • Propose (30k crawled) SQL-to-text and (crawled) table-to-text models to produce large scale synthetic datasets with enough quality;

Grappa (2021)	✅				TTS-FT, RTR-FT	• Pre-trained by column (appear in SQL) and operation prediction on 475k synthetic question-SQL pairs and masked language reconstruction on 392k table-language pairs; • Additional classification layer for column and operation prediction; • Induce grammar from annotated Text-to-SQL examples to synthesize new examples;
PICARD (2021)			✅		TTS-FT	• Constrained auto-regressive decoding by rejecting inadmissible tokens; • Inputs: token ids, predicted log-softmax scores, schema info; • Detect invalid keyword and schema item, invalid query structures, invalid table scope;
NatSQL (2021)					TTS-FT	• Propose a new SQL intermediate representation to address mismatch between NL and SQL (simplify queries: remove GROUP BY, HAVING, FROM, JOIN ON, need for nested subqueries and set operators, reduce number of schema items); • NLQ/T ⟵NN model⟶IR⟵rule⟶SQL
SmBoP (2021)			✅		TTS-FT	• Decode top K program sub-trees of a certain height at each step, learn representation of sub-program in previous step, bottom-up parsing, tree based on standard query language 'relational algebra', comparable accuracy to RAT-SQL with 2x speed-up;
Table Former (2022)		✅			RTR-FT, TableFV	• Introduce 13 types of attention biases for table-text structure (e.g., same row, header to column cell, cell to sentence); • Remove absolute row/col order in the positional embedding, add *relative positional info* by assigning 1 bias type with 1 learnable scalar; • *Bias matrix* is added to Q/K similarity before SoftMax; • Robust to row/col shuffling;
TAPEX (2022)	✅				RTR-FT, TableFV	• Pretrain only by SQL execution task; • Table source: high-quality 1,500 WTQ tables; • SQL templates from SQUALL; • SQL executor (e.g., MySQL) as supervision;

OmniTab (2022)	✅				RTR-FT	• Pretrained by masked language reconstruction using both natural and synthetic data + QA loss using TAPEX QA pairs; • Natural: use retrieval to pair Wikipedia table and NL sentences; • Synthetic: given table, use SQL sampler and SQL2NL model to generate Q&Ar; • Salient mention masking (mask shared information between table and text); • SQL2NL self-train using SQL and generated text with high OmniTab model scores;
Unified SKG (2022)					TTS-FT, RTR-FT, TableFV	• Unify 6 families 21 tasks into text-to-text format (semantic parsing, QA, data-to-text, conversational, fact verification, program-to-text);
Pasta (2022)	✅				TableFV	• Pretrain with 6 types of sentence–table cloze questions synthesized from WikiTables (Filter, Aggregation, Superlative, Comparative, Ordinal, and Unique), 1.2M pairs, 20k table<500 cells; • Use NL & SQL template to generate operation aware pre-training samples; • Choose DeBERTaV3 for its positional encoding scheme; • Preprocess table during fine-tuning (select columns containing entities linked to statement, reorder table by rows by relevance score);
TaCube (2022)		✅			RTR-FT	• Augment input table with rule-generated question-sensitive pre-computation; • Select operator types by textual mention in questions, find col/row by matching headers, and cell values with NLQ, compute all;
RegHNT  (2022)		✅	✅		RTR-FT	
RESDSQL (2023)		✅	✅		TTS-FT	• Ranking-enhanced encoder: rank and filter schema items based on additional classification logits (4 tbl, 5 col each); • Skeleton-aware decoder: generate SQL skeleton first, select value from input to fill skeleton slots; • Focal loss for tbl and col classification (referenced or not); • Execution-Guided SQL Selector;

DIN-SQL (2023)				✅	TTS-ICL	• 4-module workflow, 10 task-specific exemplars • Schema linking (identify references to schema and condition values); • Classification & decomposition (easy/non-nested complex/nested complex) for each query; • SQL generation (non-easy first generate intermediate representation then SQL); • Self-correction (assume buggy SQL ask to fix or ask to check);
Few(1) shot Table Reasoners (2023)				✅	RTR-ICL	• Truncate table with first 22 rows, first 8 cols, 10 first words in each cell; • GPT3/Codex CoT; • Findings: LLMs sometimes make simple mistakes on symbolic operations; Unable to generalize to 30+ row table; LLM prompting exhibits unpredictable randomness; far from SOTA;
BINDER (2023)				✅	TTS-ICL, TableFV	• Combine SQL and LLM, use LLM to decide which parts can be converted to SQL, unanswerable parts are replaced by LLM API; • LLM API produce values for unanswerable parts to be integrated into SQL to get candidate answers, finally vote for final answer; • Input 14 ICL exemplars, schema and first 3 rows; • *Core*: *bring in LLM for given table unentailed knowledge;*
Dater (2023)				✅	RTR-ICL	• Use LLM to decompose large table into relevant small ones (by predict row and col indexes, e.g., *col(name, cost),* row*(1,3,13)*); • Directly decompose a complex question fall into hallucination => generate abstract sub-question with masked value, convert abstract logic into SQL queries, execute SQL on decomposed table, backfill mask in sub-question with queried value, reason on sub-table & sub-question; • *Core: tackle hallucination by only using spans from given table;*
TableGPT (2023)					RTR-ICL	