# DBVinci – towards the usage of GPT engine for processing SQL Queries

Vanessa Câmara
Retail Management System
Sidia R&D Institute
Manaus, Brazil
vanessa.camara@sidia.com

Rayol Mendonca-Neto
Retail Management System
Sidia R&D Institute
Manaus, Brazil
rayol.neto@sidia.com

André Silva
Retail Management System
Sidia R&D Institute
Manaus, Brazil
andre.fernandes@sidia.com

Luiz Cordovil-Jr
Retail Management System
Sidia R&D Institute
Manaus, Brazil
luiz.cordovil@sidia.com

## ABSTRACT

One of the goals of Natural Language Processing (NLP) is transforming sentences to output relevant information in a given context. For instance, relevant applications such as chatbots, translation systems, and sentiment analysis classifiers work that way. The advance of NLP techniques made it possible to automate complex tasks, such as converting text queries to tabular data queries, specifically SQL, to return contextualized data. Since it is crucial in many areas to interpret the data to obtain information and consider the particularities of a text-to-SQL parser, we propose a SQL processing engine whose internals are customized with natural language instructions. DBVinci is our proposed processing model which is based on OpenAI's GPT-3.5 Text-davinci-003 engine that can perform language tasks such as text-to-SQL, consistent instruction-following, and supports inserting completions within text. Our framework is on top of GPT-3.5 and decomposes complex SQL queries into a series of simple processing steps, described in natural language. DBVinci outperforms well-known text-to-SQL methods (e.g., RAT-SQL and SQLova) reaching 89.7% of execution accuracy, considering WikiSQL benchmark. We also obtain impressive performance without the need of large scale annotated dataset for fine-tuning the downstream task, by achieving 90% accuracy in zero-shot setting. Therefore, we conclude that to obtain competitive results using the Pre-trained Language Model (PLM), there is no need of the "pre-training+fine-tuning" paradigm, besides that, when employing zero-shot in the proposed method, we can achieve promising results.

## CCS CONCEPTS

• **Computing methodologies → Machine translation**.

## KEYWORDS

Text-to-SQL, GPT-3.5, PLM, zero-shot, SQL processing

**ACM Reference Format:**
Vanessa Câmara, Rayol Mendonca-Neto, André Silva, and Luiz Cordovil-Jr. 2023. DBVinci – towards the usage of GPT engine for processing SQL Queries. In *Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia '23), October 23–27, 2023, Ribeirão Preto, Brazil.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3617023.3617035

## 1 INTRODUCTION

Big data processing is an unavoidable way to produce business efficiency and develop better products and services at companies. For this reason, a huge number of data is now organized in relational databases and accessed through Structured Query Language (SQL) queries. The advance of Deep Learning facilitates database querying through Natural Language Interfaces (NLI) even for casual users [1].

NLI is a research area between Natural Language Processing (NLP) and human-computer interaction that seeks to provide means for humans to interact with computers through the use of natural language [16]. Those systems not only offer an intuitive way to explore complex datasets beyond keyword-search queries but also contribute to data democratization, which allows users to easily access data and to derive value, query explanations and query result explanations from it [8].

In this context, text-to-SQL is a subarea of NLI in continuous research given its relevance on automating the return process of tabular data and easing the efforts of human users on learning and writing SQL queries [2]. The purpose of text-to-SQL problem is to convert natural language questions to SQL queries using computational resources, more specifically, Artificial Inteligence (AI) models.

To the best of our knowledge, CodexDB [10] was the first Pre-trained Language Model (PLM) without fine-tuning to solve text-to-SQL task. It is also called a SQL processing engine (SPE). Based on Codex GPT-3, the model translates text into code whilst decomposing complex SQL queries into a series of simple processing steps, described in NL. Processing steps are enriched with user-provided instructions and descriptions of database properties.

Although there are previous methods that handle the text-to-SQL tasks using pre-trained models, there still is space for advances in this area. The OpenAI's GPT model has been constantly improved with new data and new configurations, for instance. The new version employs the OpenAI's GPT-3.5 which is a large neural network, that can do any language task with a high quality, longer output and consistent instruction-following, compared to its older releases, and also supports inserting completions withing text.

Therefore, in this paper we present DBVinci, a method based on state-of-art Pretrained Language Model architecture that is refined to meet the needs of text-to-SQL task. In our scope, we implement Text-davinci-003 (DaVinci), which is part of the basis of the OpenAI's GPT-3.5 series of models and has 175 billion parameters, making it a highly capable text generator. The power of DaVinci lies in the high ability to understand instructions and generate text [14]. We can highlight our proposal does not need the fine-tuning step implementation, despite of other related methods that employed this step [5, 11].

Our main contributions are: a) to show high performance in zero-shot setting without a large scale annotated dataset; b) reinforce that large pre-trained language models can achieve competitive performance on text-to-SQL parsing without "pre-processing+fine-tuning" paradigm [7, 12]; c) Show that we achieve competitive logical form accuracy and execution accuracy metrics comparing to well-known text-to-SQL methods (e.g. Rat-SQL and SQLova) [5, 11].

This paper is organized as follows: Section 2 presents related works; Section 3 presents the proposed method; Section 4 shows the methodology; Section 5 presents results and discussion; Section 6 presents the conclusion of this research and future work.

## 2 RELATED WORK

Previous papers inspired our work such as CodexDB [10] and has very similar purpose to ours. The enabling technology for this system is OpenAI's GPT-3.0 Codex model, a large neural network that translates natural language instructions into code. The developed framework accepts queries together with natural language instructions as input. These instructions customize the way in which queries are executed. CodexDB generates code to process queries while complying with additional instructions and decomposes complex SQL queries into sequences of simple processing steps and they are formulated in NL using corresponding text templates.

Finally, automatically generated plan steps are interleaved with user-provided instructions. The resulting text is enriched with information about the database schema and physical layout. The final text is submitted to GPT-3 Codex, as known as prompt. Using this approach as a starting point, CodexDB generates code for sample queries in a training step.

Also, it is important to cite the implementation of Deep Learning models for the task. The work of Wang et al. [11] presents unified framework, called RAT-SQL for encoding relational structure in the database schema and a given question. The model is based on the self-attention mechanism, to address schema encoding, schema linking, and feature representation within a text-to-SQL encoder. They trained the method on Spider dataset [15] and WikiSQL. For the first dataset, the model achieves an accuracy of 57.2% and for the second dataset, the model achieves 78.8% of execution accuracy.

SQLova [5] is another Natural-language-to-SQL (NL2SQL) model that achieves human performance in WikiSQL dataset. The authors demonstrate the effectiveness of the architecture by proposing a BERT-based table-aware encoder and a task-specific module on the top of the encoder. The implemented method consists of two layers: encoding layer that obtains table-aware word contextualization and NL2SQL layer that generates the SQL query from the contextualized representations. The model achieves 83.6% logical form accuracy and 89.6% execution accuracy on WikiSQL test set. The authors also explain that while BERT plays a significant role, merely attaching a sequence-to-sequence model on the top of BERT leads to a poor performance, indicating the importance of properly and carefully utilizing BERT when dealing with structured data.

In terms of comparison, although the first approach (CodexDB) is very similar to ours, the authors do not consider one of the main metrics to evaluate results in Machine Language Models (MLM) such as logical form accuracy. Instead, they use precision and accuracy and report results of experiments comparing different prompt generation methods. Beyond that, Codex models are deprecated and substituted for more proficient DaVinci engines, in terms of robustness [13]. Also, hyperparameters such as temperature are essential to deliver good operation of CodexDB prototype, which is an unnecessary technique in our case, given robustness of DaVinci engine. Notwithstanding, both researches (CodexDB and ours) emphasize the necessity of evaluating PLM models for data democratization, since common users are becoming more exposed to relational databases.

Secondly, the reason we chose RAT-SQL and SQLova to compare to our model is that both are well-known robust models to solve the task and they corroborate the point that PLMs achieve as competitive performance as those models without the pre-training and fine-tuning phases, since we achieve higher results of execution accuracy comparing to both. The first model uses fine-tune phase with Bidirectional Encoder Representation from Transformers (BERT) to achieve higher results whereas SQLova approach focus on fine-tuning BERT-based and table-aware encoding layer.

## 3 DBVINCI

In this section, we present the DBVinci text-to-SQL method. In contrast to classical approaches, our method employs Pre-trained Language Model. Figure 1 summarizes the steps that compose our approach which are discussed in the next subsections.

Initially, both the natural language questions and the table structure with its columns are preprocessed (subsection 3.1) to merge data and form a single input. After that, we create a prompt in NL that is passed to DaVinci engine to create queries based on the ones from benchmark and its respective table structure. We perform post-processing (subsection 3.2) to clean the output noise that is not part of the query. Furthermore, to compare results in execution accuracy, we get the label query and predicted one and execute against database. The results of this process are recorded separately. For both logical form accuracy and execution accuracy, we implemented a script to compare results for each query. Our experiments evaluating DBVinci focus on the key step of translating an NL query into SQL.

### 3.1 Pre-processing

In this step, we mapped each table and column of the database that were being used in the queries, to visualize the data manager (DBMS with SQLite) in each table. After this step, the data and related columns (named generically as $col_0$, $col_1$, $col_2$, etc.) could be visualized with the real name to compose the query. In the query, we replaced the generic name of columns for real target ones to execute correctly. This step is necessary because the raw queries in WikiSQL are not comparable with execution accuracy metric.

### 3.2 Post-processing

Furthermore, we did the same procedure for the output results in GPT-3.5 with the DaVinci engine. In this process, only the column names were modified. GPT tables are numbered by ID, so it was necessary to map them. In order to run the generated queries, a data cleaning was performed to remove automatically generated line breaks. To perform the calculations and obtain execution accuracy metric, a query from the dataset was executed with the corresponding columns and, after that, the results were saved in a file. The same was done for the queries generated by the pre-trained model. After this step, the files were compared to verify their equity, shown in Table 1 and sorted in alphabetical order. Furthermore, it was verified that all queries that returned empty in inferences also returned empty in label queries and the semantics of the query structure were the same in both cases. In order to obtain logical form accuracy results, we cleaned the data by removing single and double quotes, line breaks and all queries in padding to check the string match of the metric.

## 4 METHODOLOGY

In this section, we describe the setup configuration, the evaluation metrics, and the benchmark dataset for the experiments.

### 4.1 Setup

The experiments were executed on Windows 10 home with 32 GB of RAM, i7 Intel processor, 8 cores for CPU and two GPUs: Video GTX

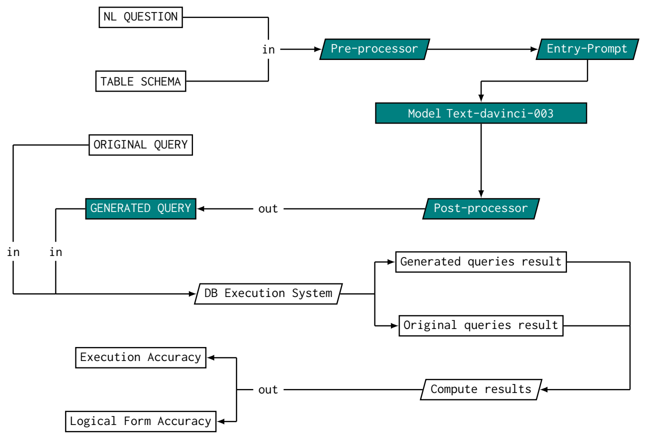

**Figure 1: DBVinci prototype evaluation process. The highlighted blocks depict the core process of the model architecture.**

1660 Ti and an integrated Intel Graphics. DBVinci is implemented in Python 3 and accesses OpenAI's GPT-3.5 Text-davinci-003 engine model via OpenAI's Python API, available at their website[1]. The experiments use the most recent version of DaVinci engine[2] with the maximum of 4,097 tokens and an estimated 175 billion parameters. The experiments consider up to the first hundred queries as OpenAI's free service is limited for outputing all predicted queries. The data on which queries are operated is stored in the CSV format.

This is a direct method of using GPT-3.5, making the comparison interesting because, to the best of our knowledge, it is the first experiment using Text-davinci-003 as engine to process queries and evaluate both logical form accuracy and execution accuracy for text-to-SQL task. Many reasons motive us to choose Text-davinci-003 engine, such as higher quality writing comparing to Text-da-vinci-002, por example; the capability of handleling with more complex instructions; and it also has better performance at longer-form content generation.

Besides that, Codex and GPT-3 models were superseded by the more powerful 3.5 released generation models. As gpt-3.5-turbo is more optimized for chat we opted to evaluate DaVinci results for the task. However, it is important to consider we have no evidence that WikiSQL was used to train OpenAi's models or not.

### 4.2 Evaluation Metrics

To evaluate the performance of our DBVinci method, we adopted logical form accuracy and execution accuracy on the benchmark since they are commonly applied to compare text-to-SQL models. Also, we emphasize that one model can have logically equivalent queries expressed by completely different strings, thus, we apply execution accuracy metric that deals with the semantics of different queries with same meaning. Let the logical form accuracy be $\mathrm{Acc}_{\mathrm{lf}} = N_{lf}/N$ and let the execution accuracy be $\mathrm{Acc}_{\mathrm{ex}} = N_{ex}/N$, where $N$ is the total number of examples, $N_{lf}$ is the number of queries that have an exact string match with the ground truth query [16], and $N_{ex}$ represents the number of queries that, when executed, produce the correct outcome [4].

Another part of the research is the evaluation in zero-shot learning for the model in order to perform a self-question-answer in text-to-SQL task. For the evaluation, the prompts are not part of the WikiSQL training data to the model and they were based on possible business necessities. We verified that the model can generate a well-desired result for it. This process was done for a maximum of 100 NL queries and we evaluated accuracy for it as follows. We divide the number of syntactically correct predicted queries by the total number of queries predicted by the model for all predicted queries.

### 4.3 Dataset

In our experiments we employed *WikiSQL*. This is one of the first large-scale, multi-domain benchmarks that made it possible to train and evaluate neural text-to-SQL systems. Wiki SQL was developed by Zhong et al. [16]. The dataset has 80,654 hand-annotated examples of questions and SQL queries distributed across 24,241 tables

[1]https://platform.openai.com/docs/models/moderation
[2]https://platform.openai.com/docs/models/gpt-3-5

from Wikipedia that is an order of magnitude larger than comparable datasets. It is ideal to train models that convert SQL to a single table, and also it provides train/dev/test splits such that each table is only in one split. This requires models to generalize not only new questions but also to new table schemas.

## 5 RESULTS AND DISCUSSION

The following experiments compare DBVinci to well-known baselines that translate NL queries to SQL. To perform the experiments, it is required a text-to-SQL benchmark that features natural language questions, along with corresponding queries. In our case the WikiSQL. [6:59 PM] Luiz Alberto Queiroz Cordovil Junior

**Table 1: DBVinci performance on WikiSQL compared to other approaches. "LF Acc" = Logical Form Accuracy; "Ex. Acc" = Execution Accuracy.**

| Model | $\text{Acc}_{\text{lf}}$ (%) | $\text{Acc}_{\text{ex}}$ (%) |
|---|---|---|
| Coarse2Fine [3] | 71.7 | 78.5 |
| DBVinci (ours) | 62.0 | **89.7** |
| HybridNet [6] | 83.8 | 89.2 |
| IncSQL [9] | 49.9 | 83.7 |
| RAT-SQL [11] | 73.3 | 78.8 |
| Seq2SQL [16] | 48.3 | 59.4 |
| SQLova [5] | **83.6** | 89.6 |

Corresponding results for the WikiSQL benchmark are available with recent methods achieving 62% of logical form accuracy ($\text{Acc}_{\text{lf}}$), 89.7% of execution accuracy ($\text{Acc}_{\text{ex}}$), which are the main metrics to evaluate most frameworks. In Table 1, we show $\text{Acc}_{\text{lf}}$ and $\text{Acc}_{\text{ex}}$ on the WikiSQL test set for DBVinci and compare to other approaches. Execution accuracy of DBVinci shows competitiveness when compared to other methods that are fine-tuned (IncSQL, RAT-SQL and SQLova) and, surprisingly, even better than other non fine-tuned models (Coarse2Fine, HybridNet, Seq2SQL).

One can observe that the string matching between predicted and labeled queries differ in 38% of the evaluated NL queries. However, execution accuracy tends to perform higher results than logical form because, even if label query and predicted query are not equal strings (syntactically different queries), they carry the same semantic meaning, returning the same registers and columns of tables from the query. That tends to happen because our SQL processing engine often generates query prediction with `LIKE` clauses in queries. Furthermore, we observed that there was no false positives in the execution accuracy. A false positive may occur when two queries return the same result, but are semantically different (e.g., when they return aggregation of two different columns that accidentally yield the same value).

For zero-shot metric, the generated queries were manually validated, since the labels are unseen and there is no benchmark to compare queries. Table 2 presents examples of predicted SQL queries generated by DBVinci. The queries in NL were also predicted by the model. Also, to perform zero-shot, we consider inference of NL queries unseen from the dataset and also generated NL queries that we asked the model to produce and, from those, we asked the model to convert them to SQL queries. We achieved 90.0% of accuracy in this zero-shot experiment. Even though CodexDB was not part of Table 1, it reports 22% of accuracy in zero-shot learning approach. So, it is possible to conclude that we performed well in this metric.

## 6 CONCLUSION AND FUTURE WORK

In this work, we made experiments with PLM GPT-3.5 model using DaVinci in order to solve the text-to-SQL task. The proposed model shows high performance in zero-shot setting without a task-specific annotated training data. Furthermore, different from most well-known models, we do not perform pre-training nor fine-tuning. By employing the engine of DaVinci from GPT 3.5 version as a text-to-SQL parsing, we obtained competitive results with related methods. In this approach, our results are relevant since our SQL processing engine achieves higher results of execution accuracy compared to RAT-SQL and SQLova. Both methods implement fine-tuned PLMs in their architectures.

In future research, we intend to explore the power of non fine-tuned PLMs in the same family of GPT, such as GPT-Neo and GPT-J. Altough DaVinci engine is newer and has more parameters compared to those ones, the weights of those models are available to explore NLP tasks, which makes the architectures more transparent to implement compared to the method in this research. Our goal of implementing these architectures for the text-to-SQL task is to investigate the trade-off behavior of having a more transparent model and having a more robust model in order to find the best text-to-SQL solution within the GPT family without the fine-tuning paradigm.

**Table 2: Examples of predicted SQL queries generated by DBVinci given NL queries in a zero-shot setting.**

| Natural Language Query | Predicted SQL Query |
|---|---|
| What is the number of managers with salary less than 5,000? | `SELECT COUNT(*) FROM managers WHERE salary <5,000;` |
| Who are the employees of sector B? | `SELECT * FROM employees WHERE sector = 'B';` |
| How many cell phones were sold in March? | `SELECT COUNT(*) FROM cellphones WHERE date_sold BETWEEN '2020-03-01' AND '2020-03-31';` |
| Say who are the employees with the highest amount of sales? | `SELECT employee_name, SUM(sale_value) as sales_amount FROM employees INNER JOIN sales ON employee.idemployee=sales.id_employee GROUP BY employee_name ORDER BY sales_amount DESC;` |
| Which products are available in store X? | `SELECT product_name FROM products WHERE store = 'X';` |

## REFERENCES

[1] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal* 28 (2019), 793–819.

[2] Ruichu Cai, Boyan Xu, Xiaoyan Yang, Zhenjie Zhang, Zijian Li, and Zhihao Liang. 2018. An Encoder-Decoder Framework Translating Natural Language to Database Queries. arXiv:1711.06061 [cs.CL]

[3] Amir Erfan Eshratifar, David Eigen, Michael Gormish, and Massoud Pedram. 2021. Coarse2Fine: a two-stage training method for fine-grained visual classification. *Machine Vision and Applications* 32, 2 (2021), 49.

[4] Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-SQL: reinforce schema representation with context. arXiv:1908.08113 [cs.CL]

[5] Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization. arXiv:1902.01069 [cs.CL]

[6] Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid Ranking Network for Text-to-SQL. arXiv:2008.04759 [cs.CL]

[7] Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the Text-to-SQL Capabilities of Large Language Models. arXiv:2204.00498 [cs.CL]

[8] Jaydeep Sen, Chuan Lei, Abdul Quamar, Fatma Özcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. 2020. Athena++ natural language querying for complex nested sql queries. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2747–2759.

[9] Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018. IncSQL: Training Incremental Text-to-SQL Parsers with Non-Deterministic Oracles. arXiv:1809.05054 [cs.CL]

[10] Immanuel Trummer. 2022. CodexDB: Synthesizing code for query processing from natural language instructions using GPT-3 Codex. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2921–2928.

[11] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2021. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. arXiv:1911.04942 [cs.CL]

[12] Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models. arXiv:2201.05966 [cs.CL]

[13] Prateek Yadav, Qing Sun, Hantian Ding, Xiaopeng Li, Dejiao Zhang, Ming Tan, Xiaofei Ma, Parminder Bhatia, Ramesh Nallapati, Murali Krishna Ramanathan, et al. 2023. Exploring Continual Learning for Code Generation Models. *arXiv preprint arXiv:2307.02435* (2023).

[14] Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, et al. 2023. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. *arXiv preprint arXiv:2303.10420* (2023).

[15] 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 the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3911–3921.

[16] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. arXiv:1709.00103 [cs.CL]