# Improving Column Type Annotation Using Large Language Models

Amir Babamahmoudi
babamahm@ualberta.ca
University of Alberta
Edmonton, Alberta, Canada

Davood Rafiei
drafiei@ualberta.ca
University of Alberta
Edmonton, Alberta, Canada

Mario A. Nascimento
m.nascimento@northeastern.edu
Northeastern University
Vancouver, British Columbia, Canada

## ABSTRACT

Large Language Models (LLMs) have demonstrated exceptional potential in a variety of tasks, including question answering, natural language to SQL conversion, and tabular data processing. However, their potential for Column Type Annotation (CTA)—a crucial component of data integration and analysis—remains largely unexplored. This paper investigates whether advanced prompting techniques can improve LLM performance on CTA tasks and examines the design of such techniques. In particular, we evaluate two reasoning-based techniques, Chain of Thought (CoT) and Retrieval-Augmented Generation (RAG), in two experimental setups: (1) single-column annotation and (2) multi-column annotation with contextual table information. Using both proprietary and open-source LLMs for a comprehensive assessment, our results reveal 10–20% performance improvements compared to traditional and previous prompting-based methods.

**VLDB Workshop Reference Format:**
Amir Babamahmoudi, Davood Rafiei, and Mario A. Nascimento. Improving Column Type Annotation Using Large Language Models. VLDB 2025 Workshop: Tabular Data Analysis (TaDA).

**VLDB Workshop Artifact Availability:**
The source code, data, and/or other artifacts have been made available at https://github.com/drafiei/llms-for-cta.

## 1 INTRODUCTION

A large volume of critical information is stored in tabular formats, making structured data essential across various domains such as finance, healthcare, and e-commerce. These datasets form the backbone of analysis, reporting, and decision-making processes. However, the full potential of tabular data can only be realized with clearly defined schemas and accurately labeled structures. Column Type Annotation (CTA) is the task of assigning meaningful semantic labels or types to table columns. Labels such as "Person," "Location," and "Height" are key to interpreting data accurately and are often mapped to established knowledge bases, such as DBpedia[1] or Schema.org[2]. Accurate column type annotation is critical for enabling downstream tasks such as structured querying and retrieval, schema matching in data integration, error detection and correction in data cleaning, and effective data exploration.

Methods developed for the CTA task can be broadly categorized into three main approaches: (1) Traditional deep learning methods, including models like DoDuo [9], rely on Pre-trained Language Models (PLMs) such as RoBERTa, which are fine-tuned to perform classification tasks on column embeddings [2, 9]. (2) Approaches leveraging knowledge graphs, as exemplified in the SemTab Challenge[3], use the rich information within knowledge graphs to infer the most plausible semantic type for a column. (3) Approaches using Large Language Models (LLMs), with their remarkable performance in diverse tasks such as language translation, question answering, and natural language to SQL conversion, have demonstrated potential in the CTA domain [6]. While PLM-based models such as DoDuo [9] have demonstrated strong performance, LLMs have emerged as competitive alternatives [6]. The effectiveness of LLMs, however, depends heavily on the prompting techniques employed, underscoring the importance of carefully crafting strategies to harness their full potential.

This paper explores whether the performance of LLMs on the CTA task can be improved using techniques such as Chain of Thought (COT) reasoning and Retrieval Augmented Generation (RAG). Our study is conducted under two settings: (1) single-column annotation, where the semantic type of a column is predicted independently, and (2) multi-column annotation, which leverages the contextual information of the entire table to improve prediction. In addition to evaluating the effectiveness of these strategies, this work provides insights into the factors influencing model performance, including potential limitations of the models themselves, the prompting strategies employed, and benchmark characteristics.

Our contributions are twofold: (1) an in-depth exploration of various prompting and reasoning strategies tailored to the CTA task, and (2) a comprehensive evaluation conducted using two LLM families across both single-column and multi-column table settings, demonstrating the effectiveness of the proposed approaches.

## 2 DATASETS AND MODELS

In this section, we detail the dataset and language models utilized in our experiments, along with the metrics employed for evaluation.

### 2.1 Dataset

To provide detailed analysis while managing the computational costs associated with LLMs, we selected a random sample from the challenging VizNet dataset[4] [4] for our experiments. Our sample was restricted to tables where the overlap between the training and


Proceedings of the VLDB Endowment. ISSN 2150-8097.

[1] https://www.dbpedia.org/
[2] https://schema.org/
[3] https://www.cs.ox.ac.uk/isg/challenges/sem-tab/
[4] https://paperswithcode.com/dataset/viznet-sato

test sets was less than 60%. This was due to the fact that approximately 58% of VizNet's test set columns fully overlapped with at least one training set column, and this could bias the CTA results. In fact, an evaluation of CTA benchmarks [1] showed that when considering the part of the dataset where each test set column exhibits 100% overlap with a training set column, DoDuo's micro F1 score reached 98% and the macro F1 score was 91%. However, when this overlap was less than 10%, the micro and macro F1 scores dropped significantly to 67% and 42%, respectively. This strong correlation between performance and the overlap ratio suggests potential model overfitting. Thus, by focusing on such more strict subset, we aimed to mitigate the bias that could arise due to overlapping data, thereby ensuring a more fair comparison between LLMs and DoDuo.

Given that our experiments involved both single-column and multi-column settings, we specifically selected tables containing three or more columns in order to use the same columns for both of our settings. In total, our experimental dataset comprised 90 tables with 300 columns spanning 32 distinct semantic types. For both the single-column and multi-column settings, we used the same dataset. In the single-column setting, we fed the columns to the model one by one, while in the multi-column setting, we provided the entire table to the model and predicted all columns simultaneously. The frequency of each type in the test sample is shown in Figure 1.

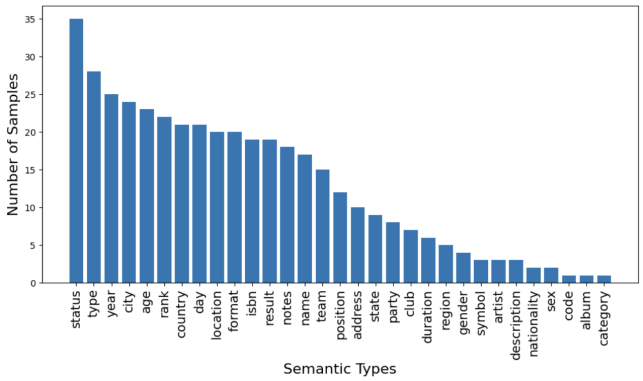


**Figure 1: Column types and type frequencies**

## 2.2 Models

For our experiments, we utilized three distinct LLMs from two model families: competitive GPT models accessed via the OpenAI API[5] and Llama 3 70B accessed through the Groq API[6]. By employing different models, we aimed to demonstrate the robustness of our approaches and ensure that our findings are not model-dependent.

# 3 IN-CONTEXT LEARNING TECHNIQUES

This section provides a detailed discussion on the design of our prompts and the adaptation of various in-context learning techniques to address CTA-related challenges.

[5]https://openai.com/
[6]https://wow.groq.com/

## 3.1 Zero-Shot Learning

To ensure a meaningful comparison with Korini and Bizer's work [6], which conducted the CTA task using LLMs, we decided to use the prompt design presented in that work as the basis of our approach. Their experiments explored various designs and formats for presenting tables and columns to the model for annotation. From these, we selected the design that achieved the best results to build upon, while also allowing us to directly compare our results with theirs.

Many LLMs, including the models evaluated here, offer a message role format that structures the prompting process as a conversation. This format distinguishes between messages from the user, the system, and the model. Figure 2 presents the general structure of our basic prompt utilized throughout this study. In the "zero-shot" setting, the model is provided with a task description along with a list of semantic type candidates representing the ground-truth labels for the columns to be annotated. This is followed by the table or column to be annotated.

**System message:** Your task is to classify the columns of a given table with only one of the following classes that are separated with commas ...

**System message:** Your instructions are: 1. Look at the columns and the types given to you. 2. Examine the values of the columns. 3. Select a type that best represents the meaning of each column. 4. Answer with the selected type only.

**User message:** Classify columns of this table:
**Column 1:** England Spain Austria Finland
**Column 2:** London Madrid Vienna Tampere

**AI message:**
**Column 1:** country
**Column 2:** city

**Figure 2: Message roles and column format in prompt design**

## 3.2 Few-Shot Learning

In this setting we provide examples along with their corresponding answers within the prompts to help the model better understand how to approach the task. Since we frame the task as a multi-class classification problem and utilize datasets with both training and test sets, we randomly selected examples from the training set to prevent introducing bias into the model. For our experiments, we chose five examples, ensuring that we stay within the model's token limitations while providing sufficient guidance for the model to perform the task effectively. This choice aligns with the approach used by the baseline [6], allowing for a direct comparison of results.

## 3.3 Retrieval-Augmented Generation (RAG)

RAG [3] has emerged as a powerful technique to enhance the capabilities of LLMs by augmenting their knowledge with external information sources. In essence, RAG combines the generative power of LLMs with the ability to retrieve relevant context from external knowledge bases, allowing them to generate more informed and

accurate responses. This method has been particularly effective in tasks requiring specialized knowledge or up-to-date information that may not be fully captured by the model's parameters [8].

When applying RAG to the CTA task, our objective is to equip the model with the most relevant in-context examples to enhance its annotation capabilitues. Observing that columns with the same or similar values often share the same or related semantic labels, we select examples from the training set that are closest in the embedding space to the table or column being annotated. Unlike traditional learning-based models that require extensive training on large datasets, LLMs offer the advantage of achieving comparable results without such task specific resource-intensive processes. Next, we provide more detail on the construction of the reference set of examples used for both single-column and multi-column RAG.

**Single-Column RAG** To build the reference set of candidates for single-column RAG, we selected a sample of 2,000 columns from the Viznet training set, ensuring a representative distribution across all semantic types. Each type was guaranteed a minimum of 10 columns to maintain adequate representation. To promote diversity and mitigate potential biases, the column selection process was randomized within each type, resulting in a balanced and diverse reference set. For embedding generation, we utilized the "text-embedding-ada-002" model provided by OpenAI[7]. The model produced vector representations that captured the conceptual semantics of the column values. The embedding process involved serializing the column values into a space-separated string, which was then passed to the model to generate the corresponding embedding vector.

**Multi-Column RAG** For the multi-column setting, we constructed our reference set by randomly selecting 300 tables from the training set, comprising a total of 1,000 columns. To prepare the tables for embedding generation, the columns within each table were serialized into a space-separated string, with columns separated by the "|" symbol. This format captured the structure and context of the table while preserving the relationships between columns. We also used the 'text-embedding-ada-002" model via the OpenAI API to generate embeddings of the serialized tables. Figure 3 illustrates an example of a serialized table, as used in the multi-column RAG process.

> Column 1: Thursday Friday Saturday | Column 2: Away Away Away | Column 3: Final Final Final | Column 4: Loss Loss Win

**Figure 3: Example of a serialized table used for RAG in the multi-column setting**

After sampling and embedding the training set, RAG for the multi-column CTA task was performed by selecting four examples for few-shot prompting [8]. These examples were chosen based on the Cosine similarity, comparing the embeddings of the sampled columns (or tables) with the embedding of the column (or table) being annotated.

[7]https://platform.openai.com/docs/guides/embeddings/embedding-models

[8]The number of in-content examples for our multi-column setting was set to 4 in order to keep the relatively larger prompt within the context window.

## 3.4 Chain of Thought Reasoning

Chain of Thought (CoT) enables LLMs to tackle problems incrementally, breaking down complex tasks into simpler, more manageable subproblems. To evaluate CoT in the context of the CTA task, we explored the idea of highlighting the relationships between columns, hence providing the model with additional context to improve its performance.

> **Example:**
> **Column 1:** Australia Brazil France Germany Italy
> **Column 2:** WIN WIN LOST WIN LOST
> **Column 3:** AUS BRA FRA GER ITA

> **Answer:**
> **Reasoning:** Based on the names of countries in the first column and the presence of results like "WIN," "TIE," and "LOST" in the second column, it appears that the first column represents teams, the second column indicates the result of a match, and the third column contains the corresponding team symbols. Therefore, the column types are ["team", "result", "symbol"].

**Figure 4: Example of CoT.**

For instance, consider a column containing the names of various countries. By examining the content of other columns in the table–such as "population," "code," and "language" in one case, or "name," "result," and "rank" in another–the model can infer whether the column's semantic type is "country" or "team." In this context, guiding the model to predict column types through CoT can improve its accuracy. However, CoT was not applied in the single-column setting, as the lack of relational context offered limited opportunity for meaningful reasoning.

To incorporate CoT into the prompt, we added a reasoning step before the answer for each example provided to the model in the few-shot setting, and for each prompt we used 5 examples, allowing for a direct comparison with simple few-shot, and the RAG method. This encourages the model to engage in a similar reasoning process before predicting the type of a column. Figure 4 illustrates an example of this approach, including the reasoning step.

# 4 EVALUATION RESULTS

This section evaluates our LLM-based approaches leveraging two GPT-based models and Llama 3, and compares their performance with DoDuo, a state-of-the-art learning-based model.

## 4.1 Single-Column Setting

Tables 1 shows the results of Llama 3, GPT-3.5-turbo and the newer GPT-4.1-mini, using zero-shot, few-shot, and RAG techniques, alongside the results of DoDuo in the single-column setting. In this setup, the prompts asked the model to predict only one column at a time, without any additional context from the table from which the column was selected.

The results reveal that RAG is the most effective prompting technique across all three models, consistently improving their performance in the single-column setting. All models surpassed the traditional DoDuo benchmark under RAG prompting. GPT-4.1-mini achieved the highest scores (micro F1: 0.81, macro F1: 0.52), with

**Table 1: Macro and Micro F1 scores for different prompting methods using using two model families for single column setting**

| Model | Method | Micro F1 | Macro F1 |
|---|---|---|---|
| - | DoDuo | 0.76 | 0.47 |
| GPT-3.5-turbo | Zero-Shot | 0.45 | 0.33 |
| GPT-3.5-turbo | Few (5-shots) | 0.50 | 0.33 |
| GPT-3.5-turbo | **RAG (5-shots)** | **0.79** | **0.49** |
| GPT-4.1-mini | Zero-Shot | 0.73 | 0.49 |
| GPT-4.1-mini | Few-Shot (5-shots) | 0.71 | 0.52 |
| GPT-4.1-mini | **RAG (5-shot)** | **0.81** | **0.52** |
| Llama 3 | Zero-Shot | 0.66 | 0.39 |
| Llama 3 | Few (5-shots) | 0.73 | 0.48 |
| Llama 3 | **RAG (5-shots)** | **0.80** | **0.48** |

**Table 2: Macro and Micro F1 scores for different prompting methods using two model families for multi-column setting**

| Model | Method | Micro F1 | Macro F1 |
|---|---|---|---|
| - | DoDuo | 0.82 | 0.53 |
| GPT-3.5-turbo | Zero-Shot | 0.71 | 0.45 |
| GPT-3.5-turbo | Few (4-shots) | 0.74 | 0.52 |
| GPT-3.5-turbo | CoT (4-shots) | 0.79 | 0.46 |
| GPT-3.5-turbo | **RAG (4-shots)** | **0.87** | **0.65** |
| GPT-4.1-mini | Zero-Shot | 0.77 | 0.55 |
| GPT-4.1-mini | Few-Shot (4-shots) | 0.79 | 0.54 |
| GPT-4.1-mini | COT (4-shots) | 0.85 | 0.64 |
| GPT-4.1-mini | **RAG (4-shot)** | **0.88** | **0.67** |
| Llama 3 70B | Zero-Shot | 0.69 | 0.44 |
| Llama 3 70B | Few (4-shots) | 0.78 | 0.53 |
| Llama 3 70B | CoT (4-shots) | 0.84 | 0.55 |
| Llama 3 70B | **RAG (4-shots)** | **0.87** | **0.65** |

Llama 3 close behind (micro F1: 0.80, macro F1: 0.48). Both models consistently outperformed GPT-3.5-turbo, which also performed well with RAG (micro F1: 0.79, macro F1: 0.49). While Llama 3 exhibited more stable performance across all prompting strategies, showing moderate gains from few-shot prompting, GPT-3.5-turbo's results varied significantly, showing weaker performance with zero-shot and few-shot approaches. These findings underscore the importance of advanced models—such as Llama 3 and GPT-4.1-mini—for column type annotation and highlight RAG's ability to optimize predictions, particularly for weaker models like GPT-3.5-turbo that struggle under standard prompting techniques.

## 4.2 Multi-Column Setting

In contrast to the single-column experiments, the multi-column setting provides the model with additional context from the table, allowing it to leverage relationships between columns to improve its predictions. In this setup, the prompts present multiple columns simultaneously, enabling the model to draw upon the interconnected data to determine the semantic types more accurately. Table 2 displays the results of GPT-4.1-mini, GPT-3.5-turbo and Llama 3, using zero-shot, few-shot, RAG and CoT techniques, along with the results of DoDuo in the multi-column setting.

In the multi-column setting, RAG emerged as the most effective prompting technique as well, yielding identical top scores for both Llama 3 and GPT-3.5-turbo (micro F1: 0.87, macro F1: 0.65), outperforming DoDuo and [6], and a slightly better score for GPT-4.1-mini. GPT-4.1-mini again achieved the highest scores under RAG (micro F1: 0.88, macro F1: 0.67), with Llama 3 close behind (micro F1: 0.87, macro F1: 0.65), and both consistently outperformed GPT-3.5-turbo. GPT-4.1-mini and Llama 3 generally surpassed GPT-3.5-turbo across most techniques, leveraging multi-column context more effectively, particularly with CoT, where they scored micro F1: 0.84-0.85 and macro F1: 0.55-0.64. GPT-3.5-turbo, however, struggled with reasoning-based techniques, highlighting its limitations in inter-column relationship exploitation compared to Llama 3. These results reinforce trends from single-column experiments, underscoring GPT-4.1 and Llama 3's superiority for complex annotation tasks and RAG's consistent effectiveness across both settings by incorporating external knowledge to enhance predictions.

# 5 CONCLUSION

In this paper, we set out to investigate if the performance of LLMs on CTA tasks can be improved using chain-of-thought abd RAG techniques described. We compared these approaches against DoDuo and the approach presented in [6], from which we adopted our prompt design based on their best-performing strategy.

Our experiments highlighted the effectiveness of the RAG approach in both single and multi-column settings, underscoring its ability to generate relevant examples that significantly improved model performance. In the multi-column scenario, the CoT method also delivered strong results, proving to be a valuable approach when leveraging inter-column relationships.

A possible future direction for our work is applying or extending the techniques developed here to other tabular annotation tasks, such as column property annotation [7], table relationship extraction [5], and table cell classification [10].

# ACKNOWLEDGMENTS

This research was partially supported by NSERC Canada.

# REFERENCES

[1] Amir Babamahmoudi, Davood Rafiei, and Mario A. Nascimento. 2025. Evaluating Column Type Annotation Models and Benchmarks. In *Companion Proc. of the Web Conference*. 854–858.
[2] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu. 2022. Turl: Table understanding through representation learning. In *ACM SIGMOD Record*. 33–40.
[3] Y. Gao, Y. Xiong, X. Gao, K. Jia, Y. Bi J. Pan, Y. Dai, M. Wang J. Sun, and H. Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).
[4] K. Hu, S. Gaikwad, M. Hulsebos, M. Bakker, E. Zgraggen, C. Hidalgo, T. Kraska, G. Li, A. Satyanarayan, and C. Demiralp. 2019. VizNet: Towards a large-scale visualization learning and benchmarking repository. In *Proc. of CHI Conf.* 1–12.
[5] D. Jinensibieke, M. Maimaiti, W. Xiao, Y. Zheng, and X. Wang. 2024. How Good are LLMs at Relation Extraction under Low-Resource Scenario? Comprehensive Evaluation. *arXiv preprint arXiv:2406.11162* (2024).
[6] Keti Korini and Christian Bizer. 2023. Column type annotation using ChatGPT. In *VLDB 23 Workshop on Tabular Data Analysis*, Vol. 3462. RWTH Aachen, 1–12.
[7] K. Korini and C. Bizer. 2024. Column Property Annotation using Large Language Models. *Proceedinds of the Semantic Web: ESWC 2024 Satellite Events*, 61–70.
[8] P. Lewis et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Proceedings of NeurIPS*, 9459–9474.
[9] S. Suhara, J. Li, Y. Li, D. Zhang, C. Demiralp, C. Chen, and W.-C. Tan. 2022. Annotating columns with pre-trained language models. In *Proceedings of the 2022 Intl. Conf. on Management of Data*. 1493–1503.
[10] Y. Sui, M. Zhou, M. Zhou, S. Han, and D. Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proc. of WSDM*. 645–654.