A Retrieval-Augmented Framework for Tabular Interpretation with Large Language Model

Mengyi Yan[1], Weilong Ren[2] ✉, Yaoshu Wang[2], and Jianxin Li[1] ✉

[1] Beihang University, China
[2] Shenzhen Institute of Computing Sciences, China
{yanmy,lijx}@act.buaa.edu.cn, {renweilong,yaoshuw}@sics.ac.cn

Abstract. Relational tables on the web hold a vast amount of knowledge, and it is critical for machine learning models to capture the semantics of these tables such that the models can achieve good performance on table interpretation tasks, such as entity linking, column type annotation and relation extraction. However, it is very challenging for ML models to process a large amount of tables and/or retrieve inter-table context information from the tables. Instead, existing works usually rely on heavily engineered features, user-defined rules or pre-training corpus. In this work, we propose a unified Retrieval-Augmented Framework for tabular interpretation with Large language model (RAFL), a novel 2-step framework for addressing the table interpretation task. RAFL first adopts a graph-enhanced model to obtain the inter-table context information by retrieving schema-similar and topic-relevant tables from a large range of corpus; RAFL then conducts tabular interpretation learning by combining a light-weighted pre-ranking model with a re-ranking-based large language model. We verify the effectiveness of RAFL through extensive evaluations on 3 tabular interpretation tasks (including entity linking, column type annotation and relation extraction), where RAFL substantially outperforms existing methods on all tasks.

1 Introduction

Table interpretation over relational web tables has become a hot research topic over past decade, since relational tables spread around the web and store a large amount of knowledge. Table interpretation aims to uncover the semantic attributes in relational tables [1], transform the tables into machine-friendly knowledge [39], and map the data in tables to nodes in knowledge graph(KG) for additional information [34]. Table interpretation plays a crucial role in various data quality applications, *e.g.,* schema matching [25], data cleaning [46], data integration [35] and KG construction [27].

However, the study of table interpretation is still in its infancy; existing approaches usually achieve moderate performance on table interpretation tasks.
Limitations of existing works. There are 3 main drawbacks of existing works for addressing the tasks of table interpretation.
Inadequate ability in handling large tables. It is common that some tables hold hundreds of columns [48]. However, pre-trained language models (PLMs) adopted

in recent works can only accept an input with a limited length and cannot directly handle large tables (*e.g.,* TABBIE [15] and DUDUO [28]).

Limited capability of retrieving and incorporating inter-table context. Relational web tables consist of various schema-free tables from different sources. However, existing works (*e.g.,* TURL [5] and TCN [34]) can only handle relational tables with known schema and topic, and cannot align and retrieve unseen related tables from a large corpus.

PLMs are hard to read tables reliably. PLMs are pre-trained on natural language texts in either *one-directional* manner or *structure-unaware* manner (*e.g.,* Bert [6] and RoBERTA [23]). However, relational tables are naturally *two-directional* (*i.e.,* row and column) and *structure-aware*, such that language models cannot be directly applied on relational tables.

Luckily, large language model (LLM) provides a powerful solution for addressing above limitations, since (1) LLM can process a longer query than traditional PLMs (maximum to 32k tokens) and can read a whole table with additional inter-table contexts; and (2) LLM is pre-trained on a variety of corpus and thus capable of reading tables reliably. However, it is non-trivial to incorporate LLM with tabular interpretation with following challenges.

Challenges of applying LLM in table interpretation.

More related tables. Retrieval-augmented generalization (RAG) [43] is the widely-used paradigm for retrieving relative context and feeding the context into quering LLM for better performance. However, RAG is hard to effectively retrieve related tables for LLM, since it only retrieves related tables based on semantic similarity, while ignoring the structure and topic similarity between tables.

Alleviation of the hallucination problem. When as a ranking model, without an effective pre-ranking strategy, LLM may suffer from hallucination problem [42] and output factual errors or incorrect predictions, since LLM is a decoder-only generative model and is hard to constrain its output range.

To address above challenges, in this paper, we propose a unified Retrieval-Augmented Framework for tabular interpretation with Large language model (RAFL). RAFL uses a graph-enhanced retrieval system to effectively retrieve related tables from a variety of schema-free web tables, and considers both structural and semantic similarity. Moreover, RAFL adopts a two-step ranking model, including a light-weighted pre-ranking model and a LLM-based re-ranking model, which enables a reliably read for large tables and alleviates hallucination issue for LLM. Furthermore, RAFL combines generalized instruction-tuning with task-specific fine-tuning for training LLM, with a small annotation budget.

Contributions. In this paper, we make the following major contributions.

- **An unified framework for tabular interpretation learning**. We introduce RAFL, an unified end-to-end tabular interpretation framework, that handles information retrieval, self-supervised annotation and ranking procedure with state-of-the-art LLM-backboned model in a reliable manner.
- **A graph-enhanced retrieval system.** To automatically retrieve related tables from a wide variety of web table corpus, we propose a graph-enhanced retrieval model that could automatically retrieve related tables and provide

self-supervised annotations, by considering both semantic and structural similarity between web tables.

- **A two-stage ranking system with LLM**. We propose a two-stage ranking model equipped with LLM, to rank candidate annotations. This model conducts task-specific fine-tuning with LLM, by effectively combining the demonstrations returned by retrieval system with the pre-ranking candidates returned by light-weighted pre-ranking models.
- **Comprehensive evaluation**. We conduct extensive experiments to evaluate the performance of RAFL on three main table interpretation tasks. RAFL outperforms a variety of state-of-the-art tabular interpretation baselines with aspect of effectiveness and robustness. In particular, RAFL consistently outperforms all non-LLM baselines with only 25% of the training data.

In addition, Section 2 reviews related works, Section 3 presents the preliminary, models and a formal problem definition, Section 4 details the proposed RAFL, Section 5 reports experimental results, and Section 6 concludes this paper.

2 Related Work

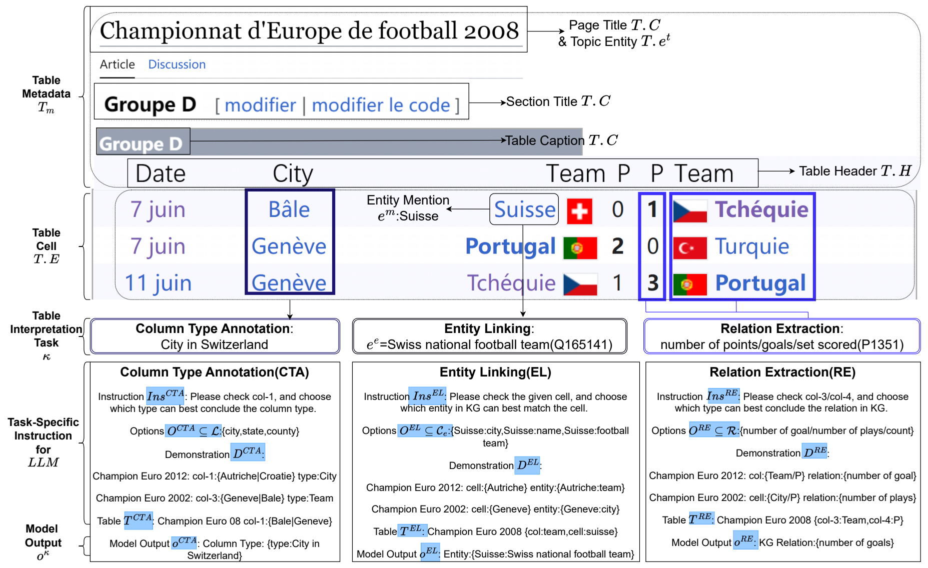
2.1 Table Interpretation

Table interpretation aims to uncover the semantics of data contained in a table, with the aim of making tabular data intelligently processable by machines [48]. This task is usually accomplished with help of existing knowledge bases. In turn, the extracted knowledge can be used for KG construction and population. There are three main tasks of table interpretation: entity linking, column type annotation and relation extraction [1, 48].

Entity linking is the task of detecting and disambiguating specific entities mentioned in a table, and is a fundamental component to many table-related tasks. Column type annotation and relation extraction both work with table columns. The former aims to annotate columns with pre-defined KG types, while the later intends to use KG relations to interpret relations between column pairs. Prior work usually incorporate such tasks with referencing KG and entity linking [49]. After linking cells with entities in KG, the types and relations in KG can be applied to annotate columns. In recent works, column annotation without the requirement of entity linking has been explored [30, 5], which modify text classification models to fit for relational tables. Relation extraction on web tables has also been studied for KG augmentation [44].

2.2 Language Models for Table Tasks

Encode-style language model for table tasks. One class of PLMs, containing BERT [6], RoBERTa [23], is pre-trained on large amount of table corpus, then employs task-specific fine-tuning on downstream tasks. A host of work,*e.g.,* TURL [5], TaBERT [39], DUDUO [28], train table-models based on encoder-stype BERT-like models, and are shown to perform well on various table interpretation tasks. However such encoder-stype table-models are lack of ability for generalizing to unseen domain or unseen tasks. Our model RAFL applies a uniform framework, and the retrieval-augmented system can incorporate few-shot demonstration for adopting to unseen tables and tasks.


Championnat d'Europe de football 2008
Page Title $T.C$ & Topic Entity $T.e^t$
Article
Discussion
Table Metadata T_m
Groupe D [modifier | modifier le code]
Section Title $T.C$
Groupe D
Table Caption $T.C$
Date
City
Team
P
P
Team
Table Header $T.H$
Table Cell $T.E$
7 juin
Bâle
Entity Mention e^m:Suisse
Suisse
0
1
Tchéquie
7 juin
Genève
Portugal
2
0
Turquie
11 juin
Genève
Tchéquie
1
3
Portugal
Table Interpretation Task κ
Column Type Annotation: City in Switzerland
Entity Linking: e^e=Swiss national football team(Q165141)
Relation Extraction: number of points/goals/set scored(P1351)
Task-Specific Instruction for LLM
Model Output o^κ

Column Type Annotation(CTA)

Instruction Ins^{CTA}: Please check col-1, and choose which type can best conclude the column type.

Options $O^{CTA} \subseteq \mathcal{L}$:{city,state,county}

Demonstration D^{CTA}:

Champion Euro 2012: col-1:{Autriche|Croatie} type:City

Champion Euro 2002: col-3:{Geneve|Bale} type:Team

Table T^{CTA}: Champion Euro 08 col-1:{Bale|Geneve}

Model Output o^{CTA}: Column Type: {type:City in Switzerland}

Entity Linking(EL)

Instruction Ins^{EL}: Please check the given cell, and choose which entity in KG can best match the cell.

Options $O^{EL} \subseteq \mathcal{C}_e$:{Suisse:city,Suisse:name,Suisse:football team}

Demonstration D^{EL}:

Champion Euro 2012: cell:{Autriche} entity:{Autriche:team}

Champion Euro 2002: cell:{Geneve} entity:{Geneve:city}

Table T^{EL}: Champion Euro 2008 {col:team,cell:suisse}

Model Output o^{EL}: Entity:{Suisse:Swiss national football team}

Relation Extraction(RE)

Instruction Ins^{RE}: Please check col-3/col-4, and choose which type can best conclude the relation in KG.

Options $O^{RE} \subseteq \mathcal{R}$:{number of goal/number of plays/count}

Demonstration D^{RE}:

Champion Euro 2012: col:{Team/P} relation:{number of goal}

Champion Euro 2002: cell:{City/P} relation:{number of plays}

Table T^{RE}: Champion Euro 2008 {col-3:Team,col-4:P}

Model Output o^{RE}: KG Relation:{number of goals}

Fig. 1: An example of a relational table from Wikipedia. The above part contains the table metadata T_m and the table cells $T.E$. The middle part illustrates examples of three table interpretation tasks κ, *i.e.,*CTA: Decide the column type for column CITY; EL: Choose the KG entity linked with cell Suisse; RE: Decide the KG relation for column pair (Team-P).The latter part presents how we organize the task-specific instruction for LLM, to provide the final output o^κ for each task. We highlight the key components for LLM instruction, and they are vital to guide LLM to output proper answer.

Decoder-style language model for table tasks. With the success of decoder-style language models (*e.g.,* GPT-3 [3] and LLaMA [31]) on few-shot or zero-shot learning tasks, pioneering researches in database field apply prompt optimization for table-tasks [45, 47, 21, 41, 29]; however, most work only fits for online LLM, and need careful human annotation and optimization. RAFL fine-tunes local LLM and can automatically retrieve and annotate examples from a large corpus, providing an affordable tool for most researchers without huge computing power.

2.3 Large Language Model

Recently, researchers find that scaling PLM (e.g., model size) often leads to an improved model capacity for models on various downstream tasks, following scaling laws [17] (*e.g.,* the 175B-parameter GPT-3 [3]). The researchers use LLM to define these large-size PLMs. Compared with PLMs, LLMs show emergent abilities [36], in-context learning [7] and instruction following [26]. These abilities guarantee LLMs to learn well on relational tables on different tasks.

3 Preliminary, Models and Problem Definition

In this section, we first introduce the preliminary and adopted models; after that, we formally define the problem of table interpretation.

3.1 Preliminary

Graph Neural Network (GNN). Consider a graph $G = (V, E, L)$, where V, E and L are the sets of vertices, edges and labels in G, respectively. GNN generates embedding for each vertex $v \in V$ by utilizing its attributes and recursively aggregating messages from 1-hop neighbors of v [10].
Pre-trained Language Model (PLM). Traditional PLMs (*e.g.,*Bert [6], GPT-2[24]) refer to the models with uniform encoder-decoder structures with 12 to 24 Transformer layers, and they have shown compelling performance on a wide range of NLP tasks [33]. PLMs are usually pre-trained on a large text corpora in the self-supervised learning manner, and then applied in multiple downstream tasks (*e.g.,* classification and regression tasks). When adopting PLMs for a specific task, one should add a task-specific layer after PLMs and fine-tune the parameters for a better performance.
Large Language Model. Large language models (LLMs), *e.g.,*GPT-3 [3] and LLaMa [31], usually refer to the decoder-only models with more than 6 billions of parameters; and they consist of 32 to 40 transformer layers [11]. LLMs are pre-trained on enormous corpora, and have been shown incredible performance on various generative tasks in the few-shot or zero-shot scenarios.

LLMs are well known for the emergent abilities [36] (*i.e.,* the sudden appearance of unseen behavior), and achieve a superior performance on unseen tasks, with no or few labeled data as demonstration. In other words, LLMs are capable of generative conversations and reasonable thinking.

However, LLMs may suffer from hallucination problem when the queried data is beyond the knowledge or ability scope of LLMs, and thus generate factual errors or unrelated answers [42]. To alleviate this problem, researchers usually adopt the following strategies to constrain the response of LLMs: (1) Instruction: a combo of prompt and options (*i.e.,* candidate outputs/answers) for guiding LLMs to accomplish a given task; (2) In-Context Learning(ICL): a method of prompt engineering that provides LLMs with demonstrations in the instruction [7]; and (3) Retrieval Augmented Generation(RAG): a method to improve the quality of LLM responses by feeding the model with retrieved relevant contextual data, without updating the parameters of LLM [20].

3.2 Models

Definition 3.1. ***(Relational Web Tables)****: A relational web table $T \in \mathcal{T}$ contains following elements: (1) Table Caption $T.C$: it includes the page title and section title (if provided) of the webpage of T; (2) Table Headers $T.H$: it refers to T' schema; (3) Topic entity $T.e^t$: it describes the topic of T; and (4) Table cells $T.E$: each cell $e \in E$ is associated with an entity pair (e^m, e^e), where e^m is the entity mention of e and e^e is the linked entity e^e of e^m in a KG $\mathcal{G}$.* □

Note that, for each cell $e \in T.E$, e^e may not always exist, when e does not contain hyperlink or the hyperlink refer to an invalid entity in $\mathcal{G}$. We simplify the notation of each element in Definition 3.1 by discarding the prefix T (*e.g.,* $T.C \to C$), when there is no ambiguity. Please refer to the above part in Figure 1 for the toy example of a relational web table from Wikipedia.

Definition 3.2. ***(Annotation):*** *For a relational web table $T \in \mathcal{T}$, we say T is (1) annotated or labeled, if its task-related annotation o^κ (*i.e., *$e^e \in \mathcal{C}_e$ for EL, $l \in \mathcal{L}$ for CTA and $r \in \mathcal{R}$ for RE) is partially or all provided by users; (2) unannotated or unlabeled, if none of O^κ is provided by users; or (3) self-annotated, if partial or all of O^κ is predicted by a model trained on T_{train}.* □

For a table T in $\mathcal{T}$, it may be associated with 3 states: annotated, unannotated and self-annotated. We denote as $\mathcal{T}_{train}$ ($\subset \mathcal{T}$) the set of annotated tables. Note that $\mathcal{L}$ are pre-defined semantic types set, $\mathcal{R}$ are pre-defined relations from an existing KG $\mathcal{G}$, and $\mathcal{C}_e$ is the candidate entities set for cell e (extracted from $\mathcal{G}$). Please check the middle part in Figure 1 for examples of $\mathcal{L}$, $\mathcal{R}$ and $\mathcal{C}_e$.

Definition 3.3. ***(Table Metadata)****: Given a relational web table T defined in Definition 3.1, its metadata $T_m = (T.C, T.H, e^t)$ contains the table caption $T.C$, headers $T.H$ and the topic entity e^t.* □

A table metadata refers to a relational web table without table cells.

Definition 3.4. ***(Related Tables)****: Given a relational web table T, its related tables $T_{related}$ contains a set of tables $T_j \in \mathcal{T}$, where the similarity score $Sim(T_j, T)$ between T and T_j is no less than a threshold γ.* □

3.3 Problem Definition

Table Interpretation. We formally define the problem of table interpretation via large language models.

Definition 3.5. ***(Table Interpretation)****: Given a relational web table $T \in \mathcal{T}$, a large language model $\mathcal{M}_G$, a knowledge graph $\mathcal{G}$, a specific task κ, the κ-related information T^κ of T, a κ-related instruction Ins^κ, a set D^κ of κ-related demonstrations (*i.e., *related tables $T_{related}$ of T) and a domain O^κ of κ-related answers, a table interpretation task adopts $\mathcal{M}_G$ to uncover the semantics contained in T and then selects an element o^κ from O^κ as the answer.* □

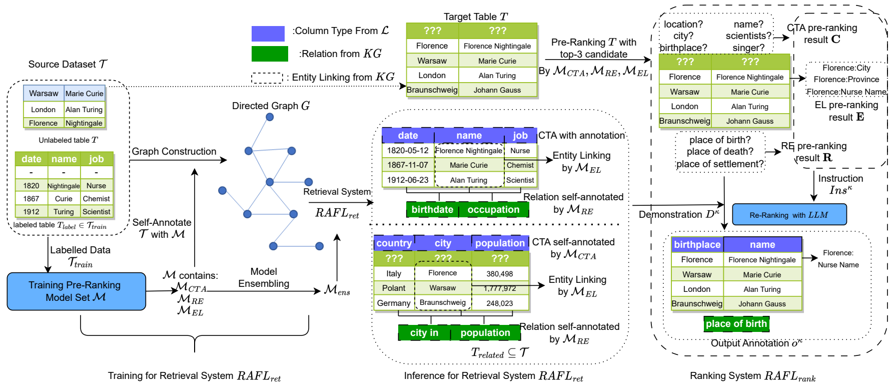
As shown in Definition 3.5, one can define an instance of a table interpretation task as a quadruple $t = (Ins^\kappa, D^\kappa, T^\kappa, O^\kappa)$. Following [1, 48], we focus on three main table interpretation tasks in this work: entity linking (EL), column type annotation (CTA) and relation extraction (RE), *i.e.*, $\kappa = \{\mathsf{EL}, \mathsf{CTA}, \mathsf{RE}\}$.

Please refer to the middle part and the caption in Figure 1 for the toy example of each specific task.

In the sequel, we detail each of the table interpretation tasks.

Entity Linking. Given a relational web table T and a knowledge graph $\mathcal{G}$, entity linking (EL) is to link the entity mention e^m of each cell $c \in E$ in T with its corresponding entity $e^{\mathcal{G}}$ in $\mathcal{G}$, where e^m and $e^{\mathcal{G}}$ refer to a same entity.

Specifically, given an entity mention e^m of a cell and a set $\mathcal{C}_e$ of potential linked entities e^e in $\mathcal{G}$, the EL task can be formulated as: $t = (Ins^{\mathsf{EL}}, D^{\mathsf{EL}}, (e^m, T_m), \mathcal{C}_e)$, where Ins^{EL} and D^{EL} are the instructions and demonstrations of the EL task respectively, $T^{\mathsf{EL}} := (e^m, T_m)$ includes the entity mention e^m of each cell $e \in E$ in T and the table metadata T_m of T as context, and $O^\kappa := \mathcal{C}_e$ limits the domain


Source Dataset $\mathcal{T}$
Unlabeled table T
labeled table $T_{label} \in \mathcal{T}_{train}$
Labelled Data $\mathcal{T}_{train}$
Training Pre-Ranking Model Set $\mathcal{M}$
$\mathcal{M}$ contains: $\mathcal{M}_{CTA}$ $\mathcal{M}_{RE}$ $\mathcal{M}_{EL}$
Model Ensembling
$\mathcal{M}_{ens}$
Self-Annotate $\mathcal{T}$ with $\mathcal{M}$
Graph Construction
Directed Graph G
Retrieval System $RAFL_{ret}$
:Column Type From $\mathcal{L}$
:Relation from KG
: Entity Linking from KG
Target Table T
Pre-Ranking T with top-3 candidate By $\mathcal{M}_{CTA}, \mathcal{M}_{RE}, \mathcal{M}_{EL}$
CTA with annotation
Entity Linking by $\mathcal{M}_{EL}$
Relation self-annotated by $\mathcal{M}_{RE}$
birthdate
occupation
CTA self-annotated by $\mathcal{M}_{CTA}$
city in
population
$T_{related} \subseteq \mathcal{T}$
location? city? birthplace?
name? scientists? singer?
CTA pre-ranking result $\mathbf{C}$
Florence:City
Florence:Province
Florence:Nurse Name
EL pre-ranking result $\mathbf{E}$
place of birth? place of death? place of settlement?
RE pre-ranking result $\mathbf{R}$
Instruction Ins^{κ}
Demonstration D^{κ}
Re-Ranking with LLM
place of birth
Florence: Nurse Name
Output Annotation o^{κ}
Training for Retrieval System $RAFL_{ret}$
Inference for Retrieval System $RAFL_{ret}$
Ranking System $RAFL_{rank}$

Warsaw	Marie Curie
London	Alan Turing
Florence	Nightingale

date	name	job
-	-	-
1820	Nightingale	Nurse
1867	Curie	Chemist
1912	Turing	Scientist

???	???
Florence	Florence Nightingale
Warsaw	Marie Curie
London	Alan Turing
Braunschweig	Johann Gauss

date	name	job
1820-05-12	Florence Nightingale	Nurse
1867-11-07	Marie Curie	Chemist
1912-06-23	Alan Turing	Scientist

country	city	population
???	???	???
Italy	Florence	380,498
Polant	Warsaw	1,777,972
Germany	Braunschweig	248,023

???	???
Florence	Florence Nightingale
Warsaw	Marie Curie
London	Alan Turing
Braunschweig	Johann Gauss

birthplace	name
Florence	Florence Nightingale
Warsaw	Marie Curie
London	Alan Turing
Braunschweig	Johann Gauss

Fig. 2: Framework of the proposed method RAFL, CTA is for column type annotation, RE is for relation extraction, and EL is for entity linking

of linked entities of e^m in $\mathcal{G}$. Please refer to the bottom middle in Figure 1 and the upper right corner in Figure 2 for a toy example.

Column Type Annotation. Given a table T and a set of pre-defined semantic types $\mathcal{L}$, column type annotation (CTA) refers to the task of annotating a column $h \in T.H$ with a semantic type $l \in \mathcal{L}$, such that all entities in column h hold the type l. Note that a column may have multiple semantic types.

Specifically, the CTA task can be formulated as: $t = (Ins^{\mathsf{CTA}}, D^{\mathsf{CTA}}, (h, T), \mathcal{L})$, where Ins^{CTA} and D^{CTA} are the instructions and demonstrations of the CTA task respectively, $T^{\mathsf{CTA}} := (h, T)$ includes the header $h \in T.H$ and table T itself as context, and $O^{\mathsf{CTA}} := \mathcal{L}$ limits the domain of candidate types for header h. Please refer to the bottom left part in Figure 1 for a toy example.

Relation Extraction. Relation extraction is the task of mapping column pairs in table T to relations in knowledge graph $\mathcal{G}$.

Specifically, the RE task can be formulated as: $t = (Ins^{\mathsf{RE}}, D^{\mathsf{RE}}, ((h_i, h_j), T), \mathcal{R})$, where Ins^{RE} and D^{RE} are the instructions and demonstrations of the RE task respectively, $T^{\mathsf{RE}} := ((h_i, h_j), T)$ contains T itself and a header pair (h_i, h_j) in T as context, and $O^{\mathsf{RE}} := \mathcal{R}$ limits the domain of candidate relations for the header pair (h_i, h_j). Please check the bottom right part of Fig. 1 for a toy example.

We provide a illustration of the above definitions and annotations in Fig. 1.

Challenges. Given a table interpretation task $t = (Ins^{\kappa}, D^{\kappa}, T^{\kappa}, O^{\kappa})$, an effective solution is usually related to three aspects: (1) a set of informative instructions in Ins^{κ}, (2) a set of instructive demonstrations in D^{κ}, and (3) a small set of candidate answers in O^{κ}. In Section 4, we propose an approach to address the table interpretation task by refining above three items.

4 Methodology

4.1 Solution Overview

In order to address the table interpretation problem in Definition 3.5, we propose a unified Retrieval-Augmented Framework with Large language model (RAFL), which applies to different table interpretation tasks. In particular, given a set of relational web table $\mathcal{T}$, an unlabeled table $T \in \mathcal{T}$ and a specific task κ, RAFL solves the table interpretation tasks in two phases: retrieve and ranking, where the retrieve phase aims to improve the quality of instructions Ins^{κ}, demonstrations D^{κ} and candidate answer set O^{κ} by retrieving related table set $T_{related}$ from $\mathcal{T}$, and the ranking phase adopts LLM to obtain the best answer from O^{κ}.

Figure 2 illustrates a toy example of applying RAFL framework for addressing three typical table interpretation tasks: entity linking (EL), column type annotation (CTA) and relation extraction (RE). In the sequel, we present the detail of the RAFL framework.

4.2 Retrieve Phase

Given an unlabeled table $T \in \mathcal{T}$, RAFL first selects the most related tables $T_{related}$ from $\mathcal{T}$ for T, and then adds self-annotations for tables in $T_{related}$. In order to achieve this, RAFL first trains two types of models: (1) a set of bi-level (column-level and cell-level) ranking models for embedding T in an unified embedding space and self-annotating the output O^{κ}, and (2) a graph structure learning model GSL for learning the unified representations of tables in $\mathcal{T}$ in an unsupervised manner. RAFL then adopts an ensemble retrieval model (by integrating above two models) for (1) pre-ranking the task result such that only top-k candidate answers with hightest probability are provided, (2) self-annotating tables in $\mathcal{T}$, and (3) obtaining the related table $T_{related}$ of T from $\mathcal{T}$.

Training. We detail the training process of the set of bi-level ranking models and the graph structure learning model mentioned above.

Bi-level ranking model $\mathcal{M}$. Given a training set $\mathcal{T}_{train}$ of annotated tables, we aim to design and fine-tune a embedding model $\mathcal{M}$, which can embed any table $T \in \mathcal{T}$ and task-specific information (*i.e.,* column type $l \in \mathcal{L}$, KG relation $r \in \mathcal{R}$ and KG entities e in knowledge graph $\mathcal{G}$) in an unified embedding space.

We construct the training data from $\mathcal{T}_{train}$. Given an annotated table $T_{label} \in \mathcal{T}_{train}$, we retrieve and serialize multiple (query,pos) training pairs from T_{label} in both column-level and cell-level, for training ranking model $\mathcal{M}$. By using contrastive learning, for a given query, we consider any elements other than pos to be negative training pairs. The detailed training pair construction is presented in Example 1. The serialized training data is denoted as $S_{train} = (S^{\mathsf{CTA}}_{train}, S^{\mathsf{RE}}_{train}, S^{\mathsf{EL}}_{train})$ for different tasks respectively.

Example 1. Consider T_{label} in Fig. 1. We denote the page title and caption title as metadata T^m_{label}. The positive pair for training should be serialized as: (1) CTA Columns-level: query: (T^m_{label},Header:City), pos: City; (2) CTA Cell-level: query: (T^m_{label},Header:Team/P), pos: number of points; (3) RE Columns-level: query: (T^m_{label},Header:City, Cell:Bale), pos: City; (4) RE Cell-level: query: (T^m_{label},Header:Team/P, Cell:Turquie/0), pos: number of points; and (5) EL Cell-level: query: (T^m_{label},Header:Team, Cell:Suisse), pos: Swiss national football team

And all the candidates not listed in pos are labelled to be negative elements.

After having the training data, we tokenize the data and pass them to a PLM model, then fine-tuning the model with the contrastive learning loss[18]:

$\min. \sum_{(q,p)} -\log \frac{\exp(\langle \boldsymbol{emb}_q, \boldsymbol{emb}_p \rangle/\theta)}{\sum_{p' \in \mathcal{P}} \exp\left(\langle \boldsymbol{emb}_q, \boldsymbol{emb}_{p'} \rangle/\theta\right)}$, where q denotes query, p denotes pos, $\boldsymbol{emb}$ means embedding, $\mathcal{P}$ is the union of all positive/negative elements, and θ is the temperature parameter.

Self-annotation. When training is finished, we obtain the task-specific model set: $\mathcal{M} = (\mathcal{M}_{\mathsf{CTA}}, \mathcal{M}_{\mathsf{RE}}, \mathcal{M}_{\mathsf{EL}})$ trained from S^{CTA}_{train}, S^{RE}_{train},S^{EL}_{train} respectively.

For $\mathcal{M}_{\mathsf{CTA}}(T)$, when given an unlabeled table $T \in \mathcal{T}$, $\mathcal{M}_{\mathsf{CTA}}$ will serialize T at both the column-level and cell-level, creating S^{CTA}_{T}. It will then encode and calculate the similarity score between the embedding of S^{CTA}_{T} and $\mathcal{L}$, and ranks all elements in $\mathcal{L}$ based on the similarity. Finally, it outputs the top-1 most similar element l' as the column type annotation for T. This pipeline is also applicable to $\mathcal{M}_{\mathsf{EL}}$ and $\mathcal{M}_{\mathsf{RE}}$. Besides, we use the model ensemble method[38], to merge the same-backbone sub-models in $\mathcal{M}$ into an ensembled model $\mathcal{M}_{ens}$, avoiding domain shift problem. $\mathcal{M}(T) = (l', r', e')$ indicates that $\mathcal{M}$ will provide the self-annotation (l', r', e') for T.

Graph structure learning model GSL. We develop a graph structure learning approach to learn the unified representations of tables in $\mathcal{T}$ in an unsupervised manner, and refine the headers H in $\mathcal{T}$, to a limited pre-defined semantic type set $\mathcal{L}$ and knowledge graph relation set $\mathcal{R}$ with the fine-tuned ranking model $\mathcal{M}$.
(1) *Graph Construction.* Given a table T with annotations (l', r', e'), we transform T into a directed subgraph $G' = (V, E, L)$. G' contains cell-level edges φ_{cell} and table-level edges φ_{table}, where φ_{cell} is the edge between cells in T, and φ_{table} is the edge between the center node e^T representing table T and all cells $e^m \in T$.

For φ_{cell}, it contains triple $(e^m_{i,j}, r', e^m_{i,j'})$, where r' is the relation edge annotated by $\mathcal{M}_{\mathsf{RE}}$ for header pair $(h_j, h_{j'})$, and $e^m_{i,j}, e^m_{i,j'}$ are nodes representing cells in i-th row, j, j'-th column from table T; For φ_{table}, it contains triple $(e^T, l', e^m_{i,j})$, where e^T is the center node representing table T, l' is the column type annotated by $\mathcal{M}_{\mathsf{CTA}}$ for column h_j, and $e^m_{i,j}$ is the node representing one cell in i-th row, j-th column from table T.

By mapping a set of headers $h \in H$ into a limited value range $\mathcal{L} \cup \mathcal{R}$ as different edge types, the schema-free $\mathcal{T}$ is pruned to a dense directed graph G.

Besides, we also add the topic-level edge $\varphi_{topic} = (e^T, r^+, e^t)$, where e^t is the topic entity representing the webpage which T belongs to, r^+ is a special-defined relation meaning belong to, and e^T is the center node representing table T. By clustering the similar topic entity e^t into a set $\{e^t\}$, we can cluster the tables with the same semantic topic into one cluster.

After constructing the subgraph G' for each $T \in \mathcal{T}$, we merge all subgraph G' for T to a unified graph G for $\mathcal{T}$.

Example 2. For table T in Fig. 1, we provide examples for the different edge types constructing G', where e^t is the topic entity node representing page Champion of Europe Football 2008, e^T is the center node representing T, r' =number of points, l' =City.

- φ_{cell}: ($e^m_{2,6}$:Turquie, r':number of goals, $e^m_{2,5}$:0)
- φ_{table}: (e^T:Champion of Europe Football 2008: Group D, l':City in Switzerland, $e_{1,2}$:Bale)
- φ_{topic}: (e^T:Group D, r^+:belong to, e^t:Champion of Europe Football 2008)

(2) *Graph Representation Learning.* After constructing G, initially we adopt a pre-trained representation learning model $\mathcal{M}_{ens}$, which is ensembled from all sub-models in $\mathcal{M}$, to generate embedding for all vertices and edges in G, such that the semantic similarity trained by $\mathcal{T}_{train}$ are more likely to be maintained.

Next, we apply CompGCN[32], a GNN-style method, to learn embedded vectors for all vertices in V. We denote $G(e)$ as the graph embedding for entity $e \in G$. Please check the left part of Figure 2, for the flowchart of training stage of our retrieval model $\mathsf{RAFL}_{\mathsf{ret}}$.

Inference. Given the well trained bi-level ranking model $\mathcal{M}$ and the graph structure learning model GSL, we develop an ensemble retrieval model $\mathsf{RAFL}_{\mathsf{ret}}$ by integrating $\mathcal{M}$ and GSL, for (a) pre-ranking the task-specific candidate answers, (b) obtaining the related table $T_{related}$ of T from $\mathcal{T}$ and (c) self-annotating tables in $T_{related}$. Please refer to the middle part of Figure 2 for the flowchart of inference stage of the retrieval model $\mathsf{RAFL}_{\mathsf{ret}}$.

Pre-ranking. Although the pre-ranking model $\mathcal{M}$ is well-trained on training set $\mathcal{T}_{train}$, it still suffers from the domain shift problem, and cannot self-annotate well on the unseen unlabelled table T. However, when loose the ranking result range to top-k, the right annotation may be within the candidate top-k set with high probability.

So for the unseen unlabelled table T, we use $\mathcal{M}$ to self-annotate the top-k candidate set, denoted as $\mathcal{C} = (\mathbf{L} \subseteq \mathcal{L}, \mathbf{R} \subseteq \mathcal{R}, \mathbf{E} \subseteq \mathcal{C}_e)$ for CTA, RE and EL tasks respectively, and $|\mathbf{L}| = |\mathbf{R}| = |\mathbf{E}| = k$.

Retrieving related tables. With the pre-ranking result $(\mathbf{L}, \mathbf{R}, \mathbf{E})$, we come back to the directed graph G and sample the related tables from G. Note that a table $T' \in \mathcal{T}$ is equivalent to the center node $e^{T'} \in G$.

Firstly, we filter G to its subgraph $G_{related}$, such that only edges in $\mathbf{L} \cup \mathbf{R}$ are allowed in $G_{related}$, and retrieve the set of tables T_{common}, which center node is contained in $G_{related}$. We obtain the top-τ most-similar tables in T_{common} as $T_{related}$, by computing and ranking the graph embedding similarity provided by G add the semantic embedding similarity provided by $\mathcal{M}_{ens}$. Since all tables $T' \in T_{related}$ are also self-annotated by $\mathcal{M}$ in graph construction procedure, $T_{related}$ is naturally suitable as the demonstration D^{κ} for re-ranking LLM model.

Example 3. In Fig. 2, the middle part $T_{related}$ shows two related tables for target table T. The selection process considers both semantic and structure similarity.

In general, we denote $\mathsf{RAFL}_{\mathsf{ret}}$ as the combination of $(G, \mathcal{M})$ as a whole graph-enhanced retrieval system, and the similarity score Sim in Definition 3.4 is provided by $\mathsf{RAFL}_{\mathsf{ret}}$. Different from existing retrieval system [43] that only consider the table-level semantic similarity, our retrieval model $\mathsf{RAFL}_{\mathsf{ret}}$ considers both semantic and structure similarity in cell-level and column-level, and tends to choose tables that share common column types, relations, entities and structures with the target table T.

4.3 Ranking Phase

Functionally speaking, RAFL applies a two-step ranking strategy for annotating the target table T. In the pre-ranking step, we adopt the trained ranking model

$\mathcal{M}$ to provide a candidate set $\mathcal{C}$ of top-k options; the detail of the pre-ranking is discussed in the retrieval phase, and we do not further discuss it. Following the In-context Learning(ICL) paradigm, in the re-ranking step, RAFL takes the candidate set $\mathcal{C}$ and the self-annotated relative table set $\mathcal{T}_{related}$ as instruction, then leverages LLM model to re-rank the options in $\mathcal{C}$ and get the final annotation for T. We provide more details of the re-ranking with LLM.

Re-ranking with LLM. As the pre-ranking stage is well illustrated in Section 4.2, we directly come the final re-ranking stage with LLM, and we denote the fine-tuned LLM as $\mathcal{M}_G$.

Inspired by the success of learn-to-rank paradigm applied in recommendation system[9], we combine $\mathcal{M}$ and LLM-based $\mathcal{M}_G$ to build up our pre-ranking and re-ranking system $\text{RAFL}_{\text{rank}}$. The pre-ranking model $\mathcal{M}$ can successfully shorten the candidate set size from $|\mathcal{L}|$ to k with considerable efficiency, but cannot perform well on the final annotation decision, w.r.t the Precision@1; LLM model is suitable for decision-making from limited candidates, but requires high-quality instruction and demonstration for T, and the training cost for LLM is large.

Back to Definition 3.5, given the training data $T_{label} \in \mathcal{T}_{train}$ and task κ, the input for training $\mathcal{M}_G$ is the quadruple $(Ins^{\kappa}, D^{\kappa}, T^{\kappa}, O^{\kappa})$, where Ins^{κ} means instruction, containing task-specific prompt and the candidate set $\mathcal{C} = (\mathbf{L}, \mathbf{R}, \mathbf{E})$ provided by $\mathcal{M}(T_{label})$; D^{κ} is the subset that randomly sampled from $T_{related}$, to avoid exceeding the maximum input token limit for LLM; T^{κ} is T_{label} without annotation; $O^{\kappa} = \mathcal{C}$. And the label for training $\mathcal{M}_G$ is o^{κ}, representing the final annotation decided by LLM, and the value for o^{κ} should be selected from the candidate set O^{κ}.

After training $\mathcal{M}_G$, for any unlabeled table $T \in \mathcal{T}$, the $\mathcal{M}_G$ can provide its final annotation o^{κ} for task κ, where the input quadruple is retrieved and annotated automatically by the retrieval model RAFL_{ret}. Please check the right part of Fig. 2, for the flowchart our ranking system $\text{RAFL}_{\text{rank}}$.

Example 4. For the lower part in Fig. 1, we provide the training pair for fine-tuning LLM. Options O^{κ} are task-specific candidate set $\mathcal{C}$, demonstration D^{κ} is sampled from $T_{related}$ with self-annotation, and output o^{κ} is selected within $\mathcal{C}$.

5 Experiment

We conducted experiments on 3 widely studied tasks of table interpretation learning: column type annotation, relation extraction and entity linking. We empirically evaluated: (1) the overall performance of the proposed framework RAFL against with the state-of-the-art baselines for each task; (2) the few-shot learning capability of RAFL; (3) the effectiveness of our retrieval-model RAFL_{ret}; (4) the ablation study of our pre-ranking model $\mathcal{M}$ and our re-ranking LLM-backboned model $\mathcal{M}_G$; and (5) the efficiency of the proposed RAFL.

5.1 Setup

Dataset We evaluated our system on 4 well-known benchmark datasets [30, 5], as shown in Table 1. For CTA task, we selected the Semtab2019 dataset[4]

Table 1: Statictics of all datasets, for CTA and RE tasks, types column are the size of pre-defined type set $|\mathcal{L}|, |\mathcal{R}|$; for EL task, avg candidates column are the average entity candidates number $|\mathcal{C}_e|$ for each mentioned cell e^m

Task Type	Datasets	types(avg candidates)	#tables	#cols	avg rows	avg cols	train/test
CTA	Semtab2019[4]	275	3045	7603	69.0	2.5	9:1
	WebTables[13]	78	32262	74141	20	2.3	4:1
EL	WikiGS-EL[5]	9.1	200018	1080994	13	5.4	25:1
RE	WikiGS-RE[5]	121	54410	65026	13	1.2	30:1

and the Webtables dataset from the VizNet corpus[13]. These datasets contained vertical relational web tables with valid semantic labels of different levels of granularity, and was widely used in recent works [28, 40, 30]. For RE task, we selected the WikiGS-RE dataset from TURL[5, 8]. The pre-defined relation set $\mathcal{R}$ were selected from Freebase[2], and annotated the ground truth by majority voting. For EL task, we selected the WikiGS-EL dataset from TURL[5] for WikiGS[8].

Experiment Settings We selected bge-large-en[37] as the backbone for the pre-ranking models $\mathcal{M}$, and Vicuna-13B[22] as the backbone of re-ranking model LLM by default. The default parameter k for pre-ranking was set to 3, and the demonstration example number budget τ was set to 4. We conducted our experiment on a single machine powered by 256GB RAM and 32 processors with Intel(R) Xeon(R) Gold 5320 CPU @2.20GHz and 4 V100 GPUs. Each experiment was run 3 times and the average is reported here.

Baselines. For CTA tasks, we compared RAFL to the following baselines. (1) Sherlock [14], which combines character-level and global statistical features with semantic multi-context analysis for vector representations of table columns. (2) TaBERT [39], a method that processes queries and table contents together, selecting three key rows for snapshots and employing BERT for column classification representations. (3) TABBIE [15], using a dual-transformer structure to encode both table columns and rows, and generating embedding for target column semantic type annotation. (4) DODUO [28], which encodes all columns in a table with a transformer structure to account for inter-table context. (5) RECA [30], an approach that aligns tables based on schema-similarity and topic relevance using a new named entity schema for inter-table context integration.

For EL and RE tasks, we compared RAFL to the state-of-the-art baseline TURL[5], which is an encoder-based BERT-like model pre-trained on 570k tables, and fine-tuned on task-specific data. Since we did not use the pre-training corpus, we compare TURL without its pre-training checkpoint, and fine-tuned on the same task-specific data WikiGS-RE and WikiGS-EL respectively. Moreover, we compared RAFL to TableLLaMA[41] (resp. GPT-4) on CTA and RE (resp. RE and EL) tasks as LLM baselines, where TableLLaMA is with 7B parameters and GPT-4 is provided with 500 examples from test set for each task due to limited budget (this is why we do not mention the training size of GPT-4).

Evaluation Measures. We used the same evaluation metrics as the baselines above. (1) For CTA and RE task, following [40], we selected two types of F_1 scores, support-weighted F_1 (Micro F_1) score and macro average F_1 (Macro F_1) score as our evaluation metrics, where Micro F_1 is the weighted average of per-

Table 2: Results of task CTA on dataset Semtab2019/WebTables

Model	Semtab2019		WebTables	
	Micro F1	Macro F1	Micro F1	Macro F1
Sherlock (100%)	0.646	0.440	0.844	0.670
TaBERT (100%)	0.768	0.413	0.896	0.650
TABBIE (100%)	0.799	0.607	0.929	0.734
DODUO (100%)	0.820	0.630	0.928	0.742
RECA(25%)	0.697	0.442	0.909	0.680
RAFL (25%)	<u>0.861</u>	<u>0.743</u>	<u>0.963</u>	<u>0.825</u>
RECA(100%)	0.853	0.674	0.937	0.783
RAFL (100%)	**0.875**	**0.766**	**0.967**	**0.834**

Table 3: Ablation study of different backbone LLM model for task CTA (resp. RE) on Semtab2019/WebTables (resp. WikiGS-RE) with 25% (resp. 10%) training data.

Model	Semtab2019		WebTables		WikiGS-RE	
	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1
TableLLaMA(7B)	0.822	0.559	0.946	0.805	0.658	0.423
RAFL (Mistral-7B)	**0.862**	0.675	0.961	0.791	0.832	0.621
RAFL (Vicuna-13B)	0.861	**0.743**	**0.963**	**0.825**	**0.893**	**0.836**

type F_1 scores (proportional to the support in each type), and Macro F_1 is the mean of all the per-type F_1 scores (treating each type equally). (2) For EL task, following [5], we adopted accuracy to evaluate the linking result, where the accuracy is the ratio of correctly linked cells to all cells.

5.2 Effectiveness Evaluation

Overall Comparison with Baselines. In this experiment, we aim to answer the following questions: (1) how is the overall performance of our method RAFL compared to various baselines? (2) is RAFL robust to few-shot learning scenario for less human-annotating cost?

We compared the performance of the RAFL with the baselines in Tables 2, 3 and 4, where the percentage in the first column indicates the ratio of $\mathcal{T}_{train}$ used for training model.

From the results, we can observe that RAFL shows outstanding few-shot learning capability, *e.g.,* with only 25% of annotated data, RAFL can persistently outperform all non-LLM baselines training with 100% of annotated data. The reasons behind are: (1) LLM is inherently suitable for few-shot scenario, and (2) our proposed retrieval model $\mathsf{RAFL}_{\mathsf{ret}}$ can maximumly improve the generalization capability of LLM, by providing LLM with high-quality demonstrations (rather than large annotated corpus).

Moreover, RAFL outperforms all baselines w.r.t. the Macro F_1, *e.g.,* by 0.201 on average. This shows that RAFL is less affected by the distribution bias in T_{train}, and can make fair classification on minority relations and types.

Ablation Study for Retrieval Model and Re-Ranking Model. In this study, we aim to answer the following questions: (1) is our retrieval-model

Table 4: Results of task RE and EL on dataset WikiGS

Model	WikiGS-RE Micro F1	WikiGS-RE Macro F1	WikiGS-EL Accuracy
TURL(10%)	0.7350	0.3088	0.6055
RAFL (10%)	<u>0.8930</u>	<u>0.8365</u>	<u>0.8705</u>
TURL(25%)	0.8601	0.6755	0.7394
RAFL (25%)	<u>0.9295</u>	<u>0.8642</u>	<u>0.8861</u>
TURL(100%)	0.9025	0.8016	0.8420
RAFL (100%)	**0.9323**	**0.9153**	**0.9112**
GPT-4	0.5295	0.4326	0.9065

Table 5: An Ablation Study on RE task

Model	Micro F1	Macro F1
RAFL w/o ret	0.3272	0.2469
RAFL w/o LLM	0.7427	0.5503
RAFL with LangChain	0.7842	0.5846
RAFL	**0.8930**	**0.8365**

$\mathsf{RAFL}_{\mathsf{ret}}$, pre-ranking model $\mathcal{M}$ and re-ranking model LLM effectiveness? (2) which component above is the most important one for RAFL?

To answer these questions, we compared RAFL with its three variants: RAFL w/o ret, RAFL w/o LLM and RAFL with LangChain, where RAFL w/o ret is the RAFL without retrieval model $\mathsf{RAFL}_{\mathsf{ret}}$ nor the demonstration D^{κ}, RAFL w/o LLM is the RAFL without the final re-ranking LLM model (*i.e.,* directly selecting the top-1 result for the pre-ranking model $\mathcal{M}$ as output), RAFL with LangChain is the RAFL by replacing the retrieval model $\mathsf{RAFL}_{\mathsf{ret}}$ by the open-source retrieval system LangChain [43] (with same hyper-parameter).

This experiment is conducted on the RE task with 10% training data. In Table 5, RAFL w/o ret suffers from the biggest performance drop, since LLM falls into severe hallucination problem, and thus cannot output correct response. RAFL w/o LLM also encounters a significant performance drop (especially in Macro F_1 score); this is because that the pre-ranking model $\mathcal{M}$ cannot use demonstration nor instruction, and can only pay attention to single column or cell. RAFL with LangChain showed moderate performance drop, since LangChain can only retrieve $T_{related}$ via semantic similarity without self-annotation.

Table 6: Experiment on Hyper-Parameter Analysis

Vary k	Micro-F1	Macro-F1	Vary τ	Micro-F1	Macro-F1
$k = 0$	0.3272	0.2469	τ=0	0.7779	0.5525
$k = 1$	0.7427	0.5503	τ=2	0.8880	0.7314
$k = 3$	**0.8930**	**0.8365**	τ=4	0.8930	**0.8365**
$k = 5$	0.8894	0.7369	τ=8	**0.9044**	0.7615

LLM **Parameter-Size Analysis**. As shown in Table 3, we evaluated the impact of parameter size of the re-ranking LLM model on the performance of RAFL. We use two models for the analysis: a 7B-parameter Mistral[16] and a 13B-parameter Vicuna model for both CTA and RE tasks. According to the scaling laws [17], a larger parameter size indicates a better performance on the model capability, as verified by the macro F1 score observed for each model. Compared to the 7B-parameter model, a 13B-parameter one improves the ability of model: it not only understands the context of downstream tasks but also performs more equitable classifications across minority relations and types. This is supported by a significant improvement in macro F1 score.

Hyper-Parameter Analysis. In this test, we evaluated the performance of RAFL by varying two key hyper-parameters: the size k of pre-ranking candidate answers and the size τ of demonstration samples. Due to space limit, we only report the results on RE task with 10% training set; the results on the CTA and EL tasks are consistent and thus omit. Note that $k = 0$ indicates that all relations in $\mathcal{R}$ are provided to LLM.

From Table 6, we can see that (1) when $k = 0$, LLM suffers from significant hallucination problem, and the performance drops drastically; (2) as k increases from 1, Micro-F_1 increases, while the Macro-F_1 may drop (*e.g.,* when $k = 5$); this indicates that more instructions may not always be suitable for LLM, since LLM may be stuck in making decision from more options; (3) as τ increases, the Micro-F_1 increases, while the Macro-F_1 may drop (*e.g.,* when $\tau = 8$); this shows that irrelevant demonstrations would mislead LLM to make wrong decision.

Table 7: Time Cost of RAFL (In Seconds)

	CTA:SimTab	CTA:WebTable	RE:WikiGS-RE	EL:WikiGS-EL
Training (25%)	3845	5010	3747	4032
Training (100%)	7321	9032	7030	8064
Inference	340	1823	450	1541

Runtime Analysis. In Table 7, we report the training and inference time of RAFL on datasets with different ratios of training data. From the results, we can see that the time cost of RAFL is scalable and acceptable, *e.g.,* no more than 1823 and 9032 seconds for training and inference, respectively, even with 100% of the training data. This is because that (1) we adopt LoRA [12] to only fine-tune a small ratio (0.106%) of parameters of LLM, and (2) we apply vLLM [19] to boost the inference speed of LLM by grouping similar queries with KV cache.

6 Conclusion

In this paper, we propose a unified retrieval-augmented framework with large language model (RAFL) for addressing the table interpretation problem, which includes the tasks of entity linking, column type annotation and relation extraction. The proposed RAFL consists of two phases: retrieve and ranking, where RAFL improves the quality of instructions, demonstrations and outputs in the retrieve phase, and then feeds these information to LLM in the ranking phase for selecting the best answer from a small-size set of high-quality candidates. The performance of the proposed approach is evaluated on extensive experiments.

7 Acknowledgments

This work was supported by Longhua Science and Technology Innovation Bureau 10162A20220720B12AB12 and NSFC 62225202.

References

[1] Bhagavatula et al. “Tabel: Entity linking in web tables”. In: *ISWC*. 2015.
[2] Bollacker et al. “Freebase: a collaboratively created graph database for structuring human knowledge”. In: *SIGMOD*. 2008.
[3] Brown et al. “Language models are few-shot learners”. In: *NIPS* (2020).
[4] Cutrona et al. “Tough tables: Carefully evaluating entity linking for tabular data”. In: *ISWC*. 2020.

[5] Deng et al. "Turl: Table understanding through representation learning". In: *ACM SIGMOD Record* (2022).
[6] Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint* (2018).
[7] Dong et al. "A survey for in-context learning". In: *arXiv preprint* (2022).
[8] Efthymiou et al. "Matching web tables with knowledge base entities: from entity lookups to entity embeddings". In: *ISWC*. 2017.
[9] Gao et al. "A survey of graph neural networks for recommender systems: Challenges, methods, and directions". In: *ACM TORS* (2023).
[10] Gilmer et al. "Neural message passing for quantum chemistry". In: *ICML*. 2017.
[11] Hoffmann et al. "Training compute-optimal large language models". In: *arXiv preprint* (2022).
[12] Hu et al. "Lora: Low-rank adaptation of large language models". In: *arXiv preprint* (2021).
[13] Hu et al. "VizNet: Towards a large-scale visualization learning and benchmarking repository". In: *CHI*. 2019.
[14] Hulsebos et al. "Sherlock: A deep learning approach to semantic data type detection". In: *ACM SIGKDD*. 2019.
[15] Iida et al. "Tabbie: Pretrained representations of tabular data". In: *arXiv preprint* (2021).
[16] Jiang et al. "Mistral 7B". In: *arXiv preprint* (2023).
[17] Kaplan et al. "Scaling laws for neural language models". In: *arXiv preprint* (2020).
[18] Karpukhin et al. "Dense passage retrieval for open-domain question answering". In: *arXiv preprint* (2020).
[19] Kwon et al. "Efficient memory management for large language model serving with pagedattention". In: *SOSP*. 2023.
[20] Lewis et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *NIPS* (2020).
[21] Li et al. "Table-gpt: Table-tuned gpt for diverse table tasks". In: *arXiv preprint* (2023).
[22] Lianmin et al. *Judging LLM-as-a-judge with MT-Bench and Chatbot Arena*. 2023.
[23] Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint* (2019).
[24] Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* (2019).
[25] Ritze et al. "Profiling the potential of web tables for augmenting cross-domain knowledge bases". In: *WWW*. 2016.
[26] Sanh et al. "Multitask Prompted Training Enables Zero-Shot Task Generalization". In: *ICLR*. 2022.
[27] Sekhavat et al. "Knowledge Base Augmentation using Tabular Data." In: *LDOW*. 2014.
[28] Suhara et al. "Annotating columns with pre-trained language models". In: *SIGMOD*. 2022.
[29] Sui et al. "Table meets llm: Can large language models understand structured table data? a benchmark and empirical study". In: *WSDM*. 2024.
[30] Sun et al. "RECA: Related Tables Enhanced Column Semantic Type Annotation Framework". In: *VLDB* (2023).
[31] Touvron et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint* (2023).
[32] Vashishth et al. "Composition-based multi-relational graph convolutional networks". In: *arXiv preprint* (2019).
[33] Vaswani et al. "Attention is all you need". In: *NIPS* (2017).
[34] Wang et al. "TCN: table convolutional network for web table interpretation". In: *WWW*. 2021.
[35] Wang et al. "Understanding tables on the web". In: *Conceptual Modeling*. 2012.
[36] Wei et al. "Emergent abilities of large language models". In: *arXiv preprint* (2022).
[37] Xiao et al. *C-Pack: Packaged Resources To Advance General Chinese Embedding*. 2023.
[38] Xiao et al. *LM-Cocktail: Resilient Tuning of Language Models via Model Merging*. 2023.
[39] Yin et al. "TaBERT: Pretraining for joint understanding of textual and tabular data". In: *arXiv preprint* (2020).
[40] Zhang et al. "Sato: Contextual semantic type detection in tables". In: *arXiv preprint* (2019).
[41] Zhang et al. "Tablellama: Towards open large generalist models for tables". In: *NAACL* (2024).
[42] Zhao et al. "A survey of large language models". In: *arXiv preprint* (2023).
[43] Harrison Chase. *LangChain*. Oct. 2022. URL: `https://github.com/langchain-ai/langchain`.
[44] Xiang Deng and Huan Sun. "Leveraging 2-hop distant supervision from table entity pairs for relation extraction". In: *arXiv preprint* (2019).
[45] Keti Korini and Christian Bizer. "Column Type Annotation using ChatGPT". In: *arXiv preprint* (2023).
[46] Mohammad Mahdavi and Ziawasch Abedjan. "Baran: Effective error correction via a unified context representation and transfer learning". In: *VLDB* (2020).
[47] Ralph Peeters and Christian Bizer. "Using ChatGPT for Entity Matching". In: *arXiv preprint* (2023).
[48] Shuo Zhang and Krisztian Balog. "Web table extraction, retrieval, and augmentation: A survey". In: *TIST* (2020).
[49] Ziqi Zhang. "Effective and efficient semantic table interpretation using tableminer+". In: *Semantic Web* (2017).