TST: A Schema-Based Top-Down and Dynamic-Aware Agent of Text-to-Table Tasks

Peiwen Jiang[1], **Haitong Jiang**[2], **Ruhui Ma**[1†], **Yvonne Jie Chen**[3†], **Jinhua Cheng**[4†],
[1]School of Computer Science, Shanghai Jiao Tong University, Shanghai, China
[2]College of Comuputer Science and Software Engineering, Shenzhen University, Shenzhen, China
[3]School of Entrepreneurship and Management, ShanghaiTech University, Shanghai, China
[4]KoGuan School of Law , Shanghai Jiao Tong University, Shanghai, China
{wayneroaming}@sjtu.edu.cn

Abstract

As a bridge between natural texts and information systems like structured storage, statistical analysis, retrieving, and recommendation, the text-to-table task has received widespread attention recently. Existing researches have gone through a paradigm shift from traditional bottom-up IE (Information Extraction) to top-down LLMs-based question answering with RAG (Retrieval-Augmented Generation). Furthermore, these methods mainly adopt end-to-end models or use multi-stage pipelines to extract text content based on static table structures. However, they neglect to deal with precise inner-document evidence extraction and dynamic information such as multiple entities and events, which can not be defined in static table head format and are very common in natural texts.

To address this issue, we propose a two-stage dynamic content extraction agent framework called TST (**T**ext-**S**chema-**T**able), which uses type recognition methods to extract context evidences with the conduction of domain schema sequentially. Based on the evidence, firstly we quantify the total instances of each dynamic object and then extract them with ordered numerical prompts. Through extensive comparisons with existing methods across different datasets, our extraction framework exhibits state-of-the-art (SOTA) performance. Our codes are available at https://github.com/jiangpw41/TST.

1 Introduction

Human natural language texts that widely exist in traditional publishing and the Internet, are the most natural and extensive forms of organizing information. How to utilize these unstructured or semi-structured text data has become one of the main challenges of contemporary information systems, such as business intelligence (Vidal-García et al., 2019), recommendation systems (Vidal-García et al., 2019), intelligent law and economics (Ash and Hansen, 2023; Ashley, 2017), and smart healthcare (Ahad et al., 2024). These practical fields require organizing text into structured forms, such as triplets, graphs, and tables, for statistical analysis and storage purposes (Jiang et al., 2024). Due to the widespread use of relational databases and the need for statistical convenience, text-to-table tasks have received widespread attention (Deng et al., 2024).

The text-to-table task aims to systematically extract key information from natural texts, often consisting of row names, column names, and cell content in triplets. With the development of deep learning techniques, especially LLMs (Large Language Models), existing research has undergone a transition from end-to-end to multi-stage. **The former** follows the ideas of early IE, such as NER (Named Entity Recognition) and RE (Relationship Extraction), modifying the Transformer network to generate tables in the form of Seq2Seq (Wu et al., 2021; Li et al., 2023; Pietruszka et al., 2022). **The latter** uses LLMs as universal solvers to extract table elements in the form of Q&A (Question and Answer), with original texts as context (Deng et al., 2024; Sundar et al., 2024). **In addition**, some methods assume that the row and column names (table schema) are unknown, and the header elements need to be generated without any external information about the task domain (Chen et al., 2024; Zhao et al., 2024b; Wu et al., 2021). Other methods assume that the table schema in the specific domain should be pre-defined for the actual demand (Dong et al., 2024; Jiang et al., 2024).

However, existing methods have obvious shortcomings. **Firstly**, the core difference between text-to-table and traditional IE is that it does not only extract elements based on linguistic concepts but

†Co-corresponding authors.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 16951–16966

rather uses domain knowledge to purposefully extract for downstream needs, which makes it difficult to generate usable structured data without external knowledge (Jiang et al., 2024). **Secondly**, the end-to-end method relies on a large amount of training data (Xu et al., 2024) and cannot handle complex table structures (Dagdelen et al., 2024). **Thirdly**, due to their focus on single documents, multi-stage methods based on LLMs either incorporate prompts with the entire document or partial chunks, or simply adapt RAG methods that are better suited for document sets (Fan et al., 2024). However, these methods often underperform in fine-grained and word-order-sensitivity tasks. **Fourthly**, most existing datasets are synthetic, simple, and have rigid enumeration structures (Jiang et al., 2024), and all related methods are for static tables, which cannot handle dynamic multi-instance extraction, such as multiple occurrences of certain events, multiple entities of the same type of role, etc.

Based on the above issues, we propose three core judgments for text-to-table at the current stage. **Firstly**, the generation of table schema is an inertia of the early end-to-end models, as these models need to include all the elements of the table. However, in the current multi-stage method, header generation can be an independent task, such as generating a knowledge graph and its schema from text (Chen et al., 2023; Pan et al.; Jiang et al., 2024), to ensure that the basis for extracting meets actual needs. **Secondly**, considering the context constraints and the requirement for model processing accuracy, a more fine-grained document content retrieval method should be adopted to construct prompt words more accurately and economically. **Finally**, it is necessary to consider the dynamic information in the text so that it can adapt to more complex table extraction tasks.

We propose a two-stage text-to-table framework TST based on type recognition retrieval and dynamic-aware agent. **Firstly**, the framework adopts a predefined domain knowledge graph schema as guidance, utilizing its dynamic structure to extract information top-down. **Secondly**, to avoid redundant information or loss of key information, we adopt a scalable inner-document type recognizer based on sentence-level chunk strategy, which extracts sentences in primitive order as evidence. **Finally**, our dynamic-aware agent adopts a two-stage strategy of *counting first and then extracting sequentially*. For entities, relationships, or attributes marked as dynamic in the schema, first perform instance counting and then add order information to the prompt for extraction. Through experiments on two of the most complex datasets considering dynamic instance situations, we demonstrate the superiority of our method over existing SOTA methods.

Our main contributions are as follows:

- Propose a two-stage dynamic-aware agent framework. To the best of our knowledge, this is the first method that formally addresses dynamic table structures.
- Propose a new scalable method for inner-document retrieval based on data type recognition, which can avoid redundant information caused by complete document prompts, as well as the problems of inaccurate and loss of word order of current RAG.
- Design comprehensive comparison and ablation experiments to demonstrate the superiority of our TST over existing SOTA methods, and the necessity of TST components.

2 Related Work

Text-to-table is a sub-task of IE but has more complex extraction formats and requirements. According to different usage paradigms, it can be divided into two types: end-to-end and multi-stage.

2.1 End-to-End

Early IE methods mainly adopt the Seq2Seq model with additional parameters in the Transformer mechanism to handle specific rows and columns. The earliest work (Wu et al., 2021) achieves the conversion by using the table constraint and table relation embeddings method, but has poor accuracy for long texts and rigidly penalize the difference in order between rows. This study (Li et al., 2023) improves it by demonstrating that row generation is sequence insensitive and adopting a sequence to sequence & set model. MedT2T (Zhao et al., 2024b) proposes a method for the medical field that utilizes adaptive medical numerical constraints to facilitate precise embedding. BnText2Table (Zariyat et al., 2024) develops a model specifically for the Bengali language based on BART and T5. Stable (Pietruszka et al., 2022) proposes a permutation-based decoder that effectively processes information by maximizing the expected logarithmic likelihood of the table content. These methods design

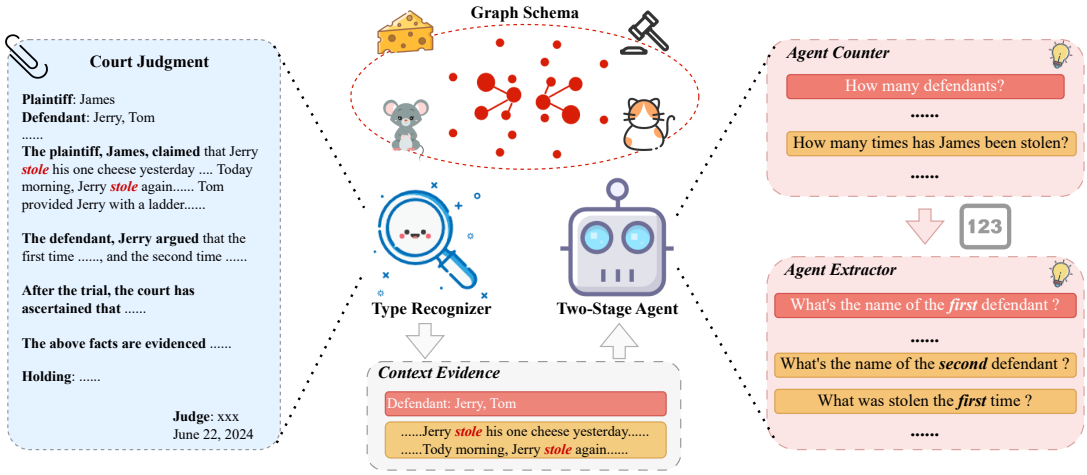

Figure 1: Overview of our dynamic-aware agent with type recognizer.

special Transformer structures for specific scenarios but rely on extensive training data and can only adapt to relatively simple static tables. With the widespread application of LLMs, LLMs become the new extractor (Dagdelen et al., 2024). This work (Coyne and Dong) proposes that LLMs can utilize their zero-shot ability as table extractors, which follows the end-to-end approach but is difficult to complete specific tasks without explicit user queries.

2.2 Multi-Stage

Mainstream researches focus on multi-stage methods based on LLMs. This work (Chen et al., 2024) proposes a new lightweight framework (DDSF), which is based on the semantic understanding ability of LLMs and extracts the header and table content separately. A similar work (Sundar et al., 2024) proposes a two-stage method called gTBLS, which firstly infers the table structure to formulate prompts for LLMs. This work (Deng et al., 2024) proposes a three-stage approach, which uses LLMs as triplet extractors and then merges triples to form a table. Although LLMs are used as general solvers in these works, they adopt the bottom-up and end-to-end manner to extract key elements and rely on simple chunked textual evidence, which can not serve precise complex table structures. OpenTE (Dong et al., 2024) designs a three-stage table extraction framework for open structures, which combines the serialization generation of BART and the few-shot ability of GPT-3.5/4. This method is based on a pre-defined table structure, but still uses an end-to-end method to generate the entire table, with LLMs only serving as correction tools. The latest work, TKGT (Jiang et al., 2024), proposes a table extraction method based on knowledge graph guidance and RAG. Although it considers the importance of pre-defined table structures for complex specialized tasks, it cannot handle dynamic table structures and can only process this dynamic information through pre-set hyper-parameters. In addition, the Hybrid-RAG it uses is suitable for a large number of document sets, often has low accuracy, and loses semantic order information between chunks when retrieving within documents (Zhao et al., 2024a).

3 Methods

As shown in Fig. 1, TST consists of three components: a predefined graph schema, a type recognizer for context retrieval, and a two-stage agent that adopts a workflow of first counting dynamic objects and then sequentially extracting them.

3.1 Predefined Graph Schema

To avoid the static limitations of traditional table schema, we adopt a dynamic graph schema as the form of field definitions, which defines the entities and relations in specific domain. As shown in the Appendix A, the schema contains dynamic structural and type information that are needed by type recognizer and agent. Specifically, for extraction tasks in specific scenarios, text sets often share an event model that includes specific roles, relationships, and attributes. For example, the Rotowire dataset (Wiseman et al., 2017) can assign various scoring data of players and the team as a whole to the attributes of the team and player roles. CPL (Jiang et al., 2024), on the other hand, natively

supports more complex graph structures of the private lending case, including roles such as the court, plaintiff, defendant, and complex lending relationships.

Algorithm 1 Dynamic-Aware Two-Stage Agent

Require: Predefined Graph Schema $Schema$
Ensure: Extracted Dynamic Field Values

$SchemaDict \leftarrow \text{EMPTYSCHEMA}(Schema)$
for all $Object \in Schema$ **do**
  if the $Object$ is dynamic **then**
    $Context \leftarrow \text{TYPEREC}(Schema, Object)$
    $ObjNumber \leftarrow \text{AGENTCOUNTER}(Schema, Object, Context)$
  else
    $ObjNumber \leftarrow 1$
  end if
  for $i = 1$ **to** $ObjNumber$ **do**
    $ObjDict \leftarrow \text{EMPTYOBJ}(Schema, Object)$
    for all $Field \in Object$ **do**
      if the $Field$ is dynamic **then**
        $ContextField \leftarrow \text{TYPEREC}(Schema, Object, i, Field)$
        $FieldNumber \leftarrow \text{AGENTCOUNTER}(Schema, Object, i, Field, ContextField)$
      else
        $FieldNumber \leftarrow 1$
      end if
      for $j = 1$ **to** $FieldNumber$ **do**
        $Value \leftarrow \text{AGENTEXTRACTOR}(Schema, Object, i, ContextField, Field, j)$
        $ObjDict[Field].append(Value)$
      end for
    end for
    $SchemaDict[Object].append(ObjDict)$
  end for
end for
return $SchemaDict$

3.2 Type Recognizer

To avoid the limitations of traditional RAG, such as the poor performance of statistically sparse retrievers in a single document and the problem of chunk word order loss in multi-channel recall and rerank, we adopt a method based on sentence-level data type recognition and supplemented by dense retrieval. This method relies on some recognizable rules in text representation, such as scoring information in Rotowire and LiveSum (Deng et al., 2024) ball games often having a specific set of terms, and lending information in court judgment documents such as CPL often including money amounts, interest rates, and dates, which can be recognized according to commonly used expression rules and have very high retrieval efficiency. In addition, this method adopts a strategy of traversing chunks, which can preserve the sentences order in the original text, providing order information for some dynamic structures. In short, our type recognizer is an improved version of Hybrid-RAG that combines type recognition keyword retriever and dense retriever.

3.3 Two-Stage Agent

Existing methods only generate once for each field, and can only use hyperparameter setting times for dynamic structural information. For example, TKGT (Jiang et al., 2024) defaults to 3 for all behaviors that may exist multiple times, which not only performs a large number of invalid searches leading to illusions but may also be unable to handle a small amount of tail information. Therefore, we propose an agent that *counts first and then extracts sequentially*. The agent adopts a top-down and dynamic-aware workflow that can deal with globally extended context and multi-instance fields. As shown in the Algorithm 1, the agent traverses the graph structure for retrieval under the guidance of a predefined schema. For dynamic multi-instance fields, we first obtain the instance number in the text and use the quantity keyword as the enhanced information for RAG. Besides, we use the type recognizer to extract the most relevant context for the current field extraction task based on the schema's elements such as entity, attribute, action, relationship, and frequency information.

Specifically, we initialize the predefined graph schema and start iterating through two layers. The first layer traverses all entity types in the schema, and the second layer traverses every field of each entity type for extraction. When traversing an entity's field, the first step is determining whether the field is static or dynamic based on the schema. If it is static, the default number is set to 1. Otherwise, the counter workflow is used to count the number of instances of the field in the document

using LLMs and RAG.

It should be noted that some fields are dependent, such as the field about the interest to be repaid as the main field, which has dependent sub-fields like interest rate, calculation start time, calculation end time, interest type, etc. We do not need to traverse all dependent fields, but should determine the instance number of the main field first to avoid invalid Q&A. When obtaining the entity, the field, its corresponding number, and other schema description information, we can more accurately retrieve relevant context and obtain answers. The core of the agent mechanism lies in a fact: for multi-instance objects, an indication about the ordinal or total number will effectively improve the performance of RAG. These will be demonstrated explicitly in the experiment section.

4 Experiment

We conduct comparative experiments with various naive methods and baselines on two datasets considering dynamic instances and conduct ablation experiments on the type recognizer component. Due to the large amount of result data, we place the result tables on the CPL dataset in the Appendix E and F.

4.1 Testbed

To explore the potential for deployment at the edge and large-scale parallelism, we focus more on small LLMs (0.5B to 8B). These models have the potential to be deployed on consumer grade GPUs, and the cost of task-oriented instruction fine-tuning is low. All small model experiments are conducted on a local server (DELL DSS8440, 8 × 3090 24GB) using Llama-Factory as the fine-tuning tool and vLLM as the inference engine. We also use OpenAI's GPT APIs to use the SOTA commercial large LLMs for comparison.

4.2 Dataset

The experiment mainly covers two datasets. **(1) LiveSum**, a recently proposed (Deng et al., 2024) synthetic football game comments dataset that aims to gather overall team data by analyzing each player's score from the comments texts. This dataset differs from previous E2E, Rotowire, and Wikipedia datasets (Wu et al., 2021) for its multi-instance and dynamic nature, indicating that a team's score on a particular indicator depends on how many players relate to that indicator score in the text. **(2) CPL**, a recently proposed (Jiang et al., 2024) lending case judgment dataset which consists of multiple roles and dynamic instance numbers, and its fields are divided into two parts (the static and dynamic). This work also proposes a method called TKGT, but it only tests on the static part previously due to inability to handle dynamic instances. In our work, we utilize the richer range of the CPL dataset dynamic part called CPL (D), whose details can be found in Appendix D.

4.3 Baseline and Model

We mainly use the previous SOTA T-Tuple-T (T^3) (Deng et al., 2024) and TKGT (Jiang et al., 2024) along with naive LLMs strategies (CoT, ICL, Fine-tuning, and GPT-4 JSON Schema) as our baselines. Besides, we choose several popular LLMs as processors to extensively test the universality of our method. For the small-scale LLMs, we use the Llama series[1], Mistral[2] and its Chinese version (Zhou and Yuqi, 2024), Qwen[3], ChatGLM3[4], Baichuan2[5], whose numbers of model weights ranges from 0.5B to 8B. As for large-scale models, we choose the GPT-4o[6].

The experiment setting can be found in Appendix C, and the prompt templates can be found in Appendix B.

4.4 Metrics

We use two metrics. **(1) The Triplet Metric**. For the experiments on CPL datasets, we use the traditional text-to-table metric consisting of precision, recall, and F1-score along with the overall error rate of the generated triplet. In addition, to further evaluate the effectiveness, all metrics above are performed at three levels: precise matching, character matching, and semantic similarity; further details of these metrics can be found at (Wu et al., 2021). **(2) The Counting Metric**. For the experiments on LiveSum datasets and our Agent Counter on CPL, we predict the number of instances of each type in each sample and evaluate RMSE and error rate (Deng et al., 2024). Based on counting tests for CPL, we select the best intermediate result to guide the generation of instance triplets.

[1] https://www.llama.com/
[2] https://mistral.ai/
[3] https://github.com/QwenLM
[4] https://github.com/THUDM/ChatGLM3
[5] https://github.com/baichuan-inc/Baichuan2
[6] https://openai.com/index/gpt-4-research/

Model	Easy		Medium		Hard		Average	
	RMSE	ER	RMSE	ER	RMSE	ER	RMSE	ER
Fine-Tuned Small LLMs								
Mistral-7B-Instruct-v0.2	1.045	**38.36**	3.832	85.11	7.115	95.52	4.564	76.03
LLaMA-2-Chat 7B	1.047	38.41	3.728	84.91	7.107	95.40	4.512	75.91
LLaMA-2-Chat 13B	**1.043**	39.22	**3.587**	**84.60**	**6.671**	**94.42**	**4.287**	**75.71**
Zero-Shot Medium and Large LLMs								
LLaMA-2-Chat 13B	0.775	33.29	4.554	87.37	5.203	93.29	4.279	75.33
LLaMA-2-Chat 13B (COT)	0.780	31.83	4.376	87.42	5.088	92.35	4.162	74.75
LLaMA-2-Chat 70B	0.410	12.34	3.189	88.59	4.941	92.41	3.455	70.48
LLaMA-2-Chat 70B (COT)	0.450	12.86	3.221	89.25	5.314	94.24	3.613	71.40
ChatGPT	0.200	8.06	2.864	72.73	4.257	90.62	3.008	61.03
ChatGPT (COT)	0.229	10.61	2.809	72.75	4.087	90.38	2.911	61.62
Claude 2.1	1.014	10.08	2.581	63.99	4.621	90.58	3.171	57.16
Claude 2.1 (COT)	1.496	14.06	2.291	61.70	4.081	90.38	2.918	56.96
Mistral Large	**0.005**	**0.27**	2.385	52.45	2.712	84.62	2.209	47.45
Mistral Large (COT)	0.018	0.73	2.311	51.82	**2.608**	**84.08**	2.139	47.12
GPT-4	0.156	4.64	**1.167**	46.05	4.114	88.53	2.273	46.32
GPT-4 (COT)	0.154	4.38	1.173	**45.86**	3.981	88.73	2.225	**46.20**
Claude 3 Opus	0.078	2.52	1.617	51.36	3.713	88.06	2.253	48.33
Claude 3 Opus (COT)	0.040	1.59	1.642	49.60	3.265	87.86	**2.079**	47.17
***Zero-Shot Large LLMs with* T^3**								
Claude 2.1 (T^3)	0.193	8.95	1.965	44.99	2.751	72.15	2.066	42.77
Mistral Large (T^3)	0.191	8.82	1.596	42.37	2.136	69.23	1.631	40.70
GPT-4 (T^3)	**0.056**	**3.18**	0.854	25.83	1.219	46.22	0.929	25.27
Claude 3 Opus (T^3)	0.081	5.30	**0.406**	**14.79**	**0.477**	**21.29**	**0.438**	**14.04**
***Original Small LLMs with* T^3 Stages**								
T^3 Stage 1	1.190	**37.83**	**5.830**	**86.92**	**11.703**	**99.40**	**7.231**	**77.77**
T^3 Stage 2	**1.183**	38.46	6.301	89.84	12.328	99.97	7.660	79.53
T^3 Stage 3	1.196	38.30	6.279	89.94	12.312	99.93	7.646	79.53
All Zero	1.192	38.06	6.303	90.19	12.361	99.97	7.675	79.60
***Original Small LLMs with* TST**								
Pure Rule	16.579	99.14	5.766	89.16	8.410	97.25	10.320	93.68
ChatGLM3-6B	0.264	12.93	2.316	60.44	11.003	99.90	5.764	58.43
Baichuan2-7B-Chat	0.731	25.60	5.665	85.91	10.809	99.90	6.768	74.33
Qwen2.5-0.5B	0.437	14.95	0.768	32.82	6.441	88.26	3.302	42.22
Mistral-7B-Instruct-v0.2	0.129	4.57	**0.628**	**26.84**	2.729	**83.02**	**1.475**	**35.32**
Llama-2-7b-chat	0.442	15.68	0.642	27.58	**2.722**	83.32	1.516	38.54
Llama-3-8B-Instruct	**0.069**	**3.28**	0.759	33.87	3.961	94.36	2.076	41.35
Qwen1.5-7B-Chat	0.131	4.87	0.642	27.83	2.733	83.19	1.480	35.93
***Fine-Tuned Small LLMs with* TST**								
ChatGLM3-6B	0.271	8.42	4.988	77.17	10.839	99.90	6.499	65.67
Llama-3-8B-Instruct	**0.034**	1.79	**0.509**	**20.97**	1.004	51.49	**0.660**	**23.81**
Mistral-7B-Instruct-v0.2	0.037	**1.62**	0.550	22.55	**0.971**	**49.01**	0.669	23.93
Qwen2.5-0.5B	0.069	2.72	2.778	42.32	3.961	89.95	2.812	44.33
Qwen1.5-7B-Chat	0.079	3.22	0.532	21.49	0.984	50.90	0.675	24.27

Table 1: Previous T^3 results (Deng et al., 2024), new results of T^3 stages ablation, and our TST method experiments with different models on LiveSum datasets using the counting metric.

4.5 Results on LiveSum Dataset

The T^3 method related to this dataset is a three-stage pipeline based on LLMs question and answer. In the first stage, a LLM is used to generate all the triplets in the entire document. In the second stage, code is generated to process and integrate these tuples. In the third stage, the code execution results are processed and saved in CSV format. The first half of Table 1 contains the original experiments results from the T^3 (Deng et al., 2024), mainly covering the experimental results using the T^3 method on the fine-tuned open source model with weights from 7B to 13B, the results of the SOTA large commercial LLMs using the zero-shot method and T^3 method. Overall, the previous results demonstrate the potential of open-source small models in this type of task and the outstanding performance using large LLMs with T^3.

Based on this, we conduct two sets of experiments as follows.

4.5.1 Results of T^3 Stages Ablation

This part explores the performance of T^3 with open-source small LLMs (Mistral-7B-Instruct-v0.2) without fine-tuning and conduct ablation experiment for its three stages. As shown in the *Original Small LLMs with T^3 Stages* part of Table 1, we split the three-stage pipeline of T^3, process the intermediate results through simple engineering methods, and evaluate them. At the same time, we list the evaluation results with all predicted results set to zero as a baseline. **Firstly**, T^3 is almost unable to function correctly on the pre-trained 7B open-source model with results similar to the all zero one. **Secondly**, generating all triples in the first stage is the most critical and possibly the only practical step, as the latter two stages provide little help in improving the evaluation results. **Finally**, the first stage of T^3 requires the entire text in the prompt templates, which can result in input lengths that are too long for most existing open-source small models.

4.5.2 Results of TST

We select a group of internationally renowned open-source models, with the smallest being Qwen with 0.5B and the largest being Llama with 8B weights, which can easily run on consumer-grade GPUs. Due to the fact that TST uses type recognizer is a rule-based keyword retriever, and LiveSum does provide a set of commonly used expressions during the synthesis process, there may be some overlap between the two. However, TST's rule-based keywords are derived from statistics rather than direct results provided by LiveSum. To prove this point, we first provide the count results of Pure Rule in the *Original Small LLMs with TST* part of Table 1. It can be seen that although it is slightly better than all zeros on the Medium and Hard fields, it has more than ten times the error on the Easy field.

In addition, we first conduct experiments on seven open-source models, among which Llama-3-8B-Instruction and Mistral-7B-Instruct-v0.2 achieve the best results at four difficulty degrees. Specifically, Llama-3-8B-Instruction's RMSE 0.069 and ER 3.28 on the Easy field exceed the vast majority of results obtained by the T^3 method. Moreover, Mistral-7B-Instruct-v0.2 gets an RMSE of 0.628 on the Medium field, even second only to Claude 3 Opus using T^3, which has an RMSE of 0.406. The experimental results of TST with small open-source models are efficient, as they demonstrate that we can not only achieve results comparable to SOTA and expensive models at almost free cost but also have the potential to open up practical avenues for large-scale text processing with their speed and low-cost scalability.

However, our goal is broader than this. We first select five well-performing models for fine-tuning. Due to the different goals of T^3 and TST, TST could not use the fine-tuning dataset of T^3. Therefore, we spend a small amount of human resources (about an hour for a human) to create a tiny dataset containing two samples of approximately 450 pairs. The tiny fine-tuning only cost about twenty minutes. It can be seen that the fine-tuned models show comprehensive improvement compared to before, with Llama-3-8B-Instruction getting results of RMSE 0.034 and ER 1.79, which surpass the optimal results of GPT-4 using T^3 of RMSE 0.056 and ER 3.18 on Easy fields.

The above results demonstrates that our TST method can rival or even defeat the SOTA T^3 methods on open-source small models without requiring much time for fine-tuning. In addition, it is worth noting that although our results on the Qwen2.5-0.5B small model are not outstanding in comparison, its extreme size can still perform well, proving the potential of ultra-small LLM in some information extraction tasks.

4.6 Results on CPL Dataset

For the CPL dataset, we select the dynamic fields as a sub-dataset named CPL (D), details can be found

in Appendix D. Besides, results of this section can be found in the Appendix E and F. We first use GPT-4o and its structured output function as the baseline. We simply convert the same TST schema into a format adapted to the OpenAI interface, adding rich prompts with explanations. In addition, in terms of small open-source models, we select five models that are good at handling both Chinese and English, and fine-tune three of them accordingly. Methodologically, we conduct experiments using T^3 method, static TKGT method, and dynamic TST method. Among them, TKGT itself cannot handle dynamic field types. To allow it to participate in the experiment, we make a simple modification and set it to extract three instances for all fields by default.

As shown in Table 3 at Appendix E, on the CPL (D) dataset, the T^3 method still performs poorly on the original open-source model, achieving almost zero gains in accurate output due to the one-time absorption of the entire document. Besides, the T^3 method fine-tuned with instructions can usually work, but they are far inferior to GPT-4o in triplet extraction and do not have practical value. The T^3 performance using GPT-4 achieves the best within the method on both *the first column* and *table headers*, but is generally inferior to TKGT and TST, and is far inferior to T^3 using the fine-tuned model on *data cells*. On the surface, for complex tables, the ability to generate all triples at once using SOTA commercial models is limited.

In contrast, although the TKGT method performs poorly on original open-source models, inferior to GPT-4o and the fine-tuned T^3 method, the TKGT method can achieve good performance after fine-tuning, significantly surpassing GPT-4o after fine-tuning on Chinese-Mistral-7B Instruction. However, it is worth noting that this is a TKGT method that assumes a set instance count of three. Finally, we conduct experiments on the TST, It can be seen that TST on models that are not fine-tuned can achieve better results than GPT-4o, T^3, and TKGT without fine-tuning. After fine-tuning, its accuracy can be further improved.

The above results demonstrate the superiority of the TST method. To further explore the reasons for its excellent performance, we apply the RMSE metric to count on all generated results, which demonstrates that the first stage Agent Counter method of TST outperforms existing methods in terms of accuracy in counting the number of dynamic instances. Details can be found at Appendix F.

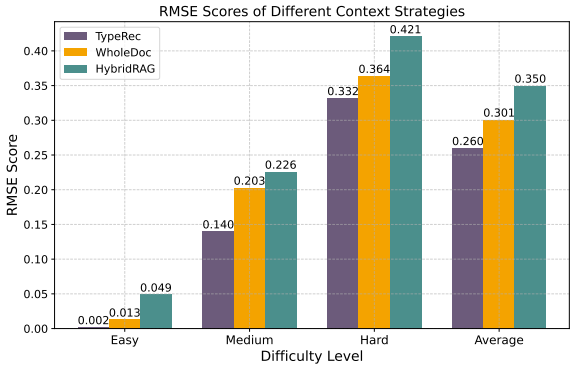

Figure 2: Results of different context strategies on CPL datasets. TypeRec refers to our method, Hybrid-RAG refers to the method used in TKGT, and WholeDoc refers to using the whole document as context.

4.7 Ablation Experiments

As shown in the Fig. 2, to validate our innovative type recognizer content retriever used in our two-stage agent method, we use the counter metric to evaluate the RMSE of different content strategies, which are all based on the best Chinese-Mistral-7B-Instruc-v0.1 model. The experimental results show that on three difficulty levels, the contextual evidence obtained by our type recognition retriever can help LLMs more accurately count the number of instances. In addition, due to the preservation of the word order of instances in the text, using the entire document as evidence has a secondary effect. However, using the Hybrid-RAG method based on dense and sparse retrievers may not be as accurate as using the entire text as evidence due to the loss of word order in multiple recalls and lower precision.

Therefore, an important finding is that for inner-document tasks such as text-to-table, the first consideration is whether a type recognizer can be used. If some of the fields do not have significant type rules, but the document length does not exceed the model input limit, consider using full-text or rule paragraphs; Hybrid-RAG can only be used as a last resort, and it is recommended to keep the original order of the chunks.

5 Conclusion

We propose a two-stage dynamic-aware agent framework, termed TST, which leverages type recognition to address the issue of neglecting fine-grained content evidence retrieval and dynamic table structures in existing text-to-table methods. Through extensive comparative and ablation ex-

periments on different models and datasets, we demonstrate the superiority of TST over existing SOTA methods T ^{3} and TKGT and demonstrate that our type recognizer is better than existing content evidence retrieval strategies.

Our framework assumes that there is already a graph schema for a certain domain, so the generalization cost mainly lies in designing a standardized schema template for the new domain. But the good news is that we do not pursue a universal and complete knowledge graph structure. Our schema directly serves downstream task requirements, that is, target extraction fields. Therefore, for any domain task with a clear goal, generalization lies in abstracting the knowledge graph skeleton (schema) from the target field set and adding some necessary metadata for each entity and field, such as basic definition, whether it is dynamic, matching templates or term sets, etc. This should be a simple task for experts in the field.

Limitations

Although our inner-document type recognizer is significantly more fine-grained and precise as a content evidence retriever compared with existing methods, it relies on the data types defined by the task objectives and needs to be extended for specific tasks. In addition, due to the diversity of language expressions in natural texts, type recognizers sometimes still need to rely on dense retrievers, which can lead to unavoidable issues of false evidence and hallucinations.

Ethics Statement

This work does not adopt AI assistants. The two datasets we use are entirely from the MIT license open-source pre-processing results of previous work (Deng et al., 2024; Jiang et al., 2024) and they do not contain any personal privacy threats. In addition, all experiments in this work follow the expected purpose of their research. Therefore, to the best of the author's knowledge, we believe that this work will not bring any additional risks.

Acknowledgement

The work was supported by Shanghai Key Laboratory of Scalable Computing and Systems, Shanghai Municipal Science and Technology Major Project, National Key Laboratory of Ship Structural Safety, Center for Empirical Legal Studies of Shanghai Jiao Tong University, National Social Science Fund Key Project of China (23AFX002), National Natural Sciences Foundation of China (72473101), and the CUPL Data Law Lab. We also thank the reviewers for their insightful comments.

References

Abdul Ahad, Zheng Jiangbina, Mohammad Tahir, Ibraheem Shayea, Muhammad Aman Sheikh, and Faizan Rasheed. 2024. 6g and intelligent healthcare: Taxonomy, technologies, open issues and future research directions. *Internet of Things*, page 101068.

Elliott Ash and Stephen Hansen. 2023. Text algorithms in economics. *Annual Review of Economics*, 15(Volume 15, 2023):659–688.

Kevin D. Ashley. 2017. *Artificial Intelligence and Legal Analytics: New Tools for Law Practice in the Digital Age*. Cambridge University Press, Cambridge.

Chen Chen, Yufei Wang, Aixin Sun, Bing Li, and Kwok-Yan Lam. 2023. Dipping plms sauce: Bridging structure and text for effective knowledge graph completion via conditional soft prompting. *arXiv preprint arXiv:2307.01709*.

Jiarui Chen, Shuangyin Li, and Yuncheng Jiang. 2024. A decomposed-distilled sequential framework for text-to-table task with llms. In *Pacific Rim International Conference on Artificial Intelligence*, pages 403–410. Springer.

Steven Coyne and Yuyang Dong. Large language models as generalizable text-to-table systems.

John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S Rosen, Gerbrand Ceder, Kristin A Persson, and Anubhav Jain. 2024. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418.

Zheye Deng, Chunkit Chan, Weiqi Wang, Yuxi Sun, Wei Fan, Tianshi Zheng, Yauwai Yim, and Yangqiu Song. 2024. Text-tuple-table: Towards information integration in text-to-table generation via global tuple extraction. *arXiv preprint arXiv:2404.14215*.

Haoyu Dong, Mengkang Hu, Qinyu Xu, Haochen Wang, and Yue Hu. 2024. Opente: Open-structure table extraction from text. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 10306–10310. IEEE.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.

Peiwen Jiang, Xinbo Lin, Zibo Zhao, Ruhui Ma, Yvonne Chen, and Jinhua Cheng. 2024. Tkgt: Redefinition and a new way of text-to-table tasks based on real world demands and knowledge graphs augmented llms. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16112–16126.

Tong Li, Zhihao Wang, Liangying Shao, Xuling Zheng, Xiaoli Wang, and Jinsong Su. 2023. A sequence-to-sequence&set model for text-to-table generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5358–5370.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap, 2023. *arXiv preprint arXiv:2306.08302*.

Michał Pietruszka, Michał Turski, Łukasz Borchmann, Tomasz Dwojak, Gabriela Pałka, Karolina Szyndler, Dawid Jurkiewicz, and Łukasz Garncarek. 2022. Stable: Table generation framework for encoder-decoder models. *arXiv preprint arXiv:2206.04045*.

Anirudh Sundar, Christopher Richardson, and Larry Heck. 2024. gtbls: Generating tables from text by conditional question answering. *arXiv preprint arXiv:2403.14457*.

Javier Vidal-García, Marta Vidal, and Rafael Hernández Barros. 2019. *Computational Business Intelligence, Big Data, and Their Role in Business Decisions in the Age of the Internet of Things*, page 1048–1067. IGI Global.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Xueqing Wu, Jiacheng Zhang, and Hang Li. 2021. Text-to-table: A new way of information extraction. *arXiv preprint arXiv:2109.02707*.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2024. Large language models for generative information extraction: A survey. *Frontiers of Computer Science*, 18(6):186357.

Tahreema Rahman Zariyat, Fahim Irfan Ahmed, Tahsina Tajrim Oishi, and Maruf Morshed. 2024. *BnText2Table–dataset and Text-to-Table generation in Bangla*. Ph.D. thesis, Brac University.

Siyun Zhao, Yuqing Yang, Zilong Wang, Zhiyuan He, Luna K Qiu, and Lili Qiu. 2024a. Retrieval augmented generation (rag) and beyond: A comprehensive survey on how to make your llms use external data more wisely. *arXiv preprint arXiv:2409.14924*.

Wang Zhao, Dongxiao Gu, Xuejie Yang, Meihuizi Jia, Changyong Liang, Xiaoyu Wang, and Oleg Zolotarev. 2024b. Medt2t: An adaptive pointer constrain generating method for a new medical text-to-table task. *Future Generation Computer Systems*, 161:586–600.

Chen Zhou and Bai Yuqi. 2024. Chinese-mistral: An efficient and effective chinese large language model. https://github.com/THU-ESIS/Chinese-Mistral.

A Dynamic Schema

Below is an example of the schema of dynamic scenario which is shown in Fig. 1. The core difference between the dynamic and static schema is whether the entity or attribute is unique. Besides, dynamics are reflected in a table as a set of sub fields appearing in multiple rows.

```
"plaintiff": {
    "metadata": {
        "type": "entity",
        "description": "xxx",
        "unique": true
    },
    "attributes": {
        "name": {
            "description": "xxx",
            "type": "string"
        },
        "address": {
            "description": "xxx",
            "type": "string"
        }
    }
},
"defendant": {
    "metadata": {
        "type": "entity",
        "description": "xxx",
        "unique": false
    },
    "attributes": {
        "name": {
            "description": "xxx",
            "type": "string",
            "unique": true
        },
        "address": {
            "description": "xxx",
            "type": "string",
            "unique": true
        }
    }
},
"stealing": {
    "metadata": {
        "type": "edge",
        "description": "xxx",
     "direction": "defendant-plaintiff",
        "unique": false
    },
    "attributes": {
        "theft_target": {
            "description": "xxx",
            "type": "string",
            "unique": true
        },
        "date": {
            "description": "xxx",
            "type": "string",
            "unique": true,
            "format": "xxxx"
        },
        "order": {
            "description": "xxx",
            "type": "integer",
            "unique": true
        },
        "stealer": {
            "description": "xxx",
            "type": "string",
            "unique": true
        },
        "person_stolen":{
            "description": "xxx",
            "type": "string",
            "unique": true
        }
    }
}
```

B TST Prompt Template

The prompt template used by TST is dynamically constructible to ensure that each attribute field value for each entity is given a prompt that is as appropriate as possible. This dynamic prompt word template system combines the CoT principle, dynamically selects ICL examples, and performs context retrieval as accurately as possible, providing the most appropriate and accurate questioning statements. As shown in Fig. 3 and 4 below, here are two examples of our TST prompt templates. Firstly, the *Instruction* section provides background knowledge, conceptual explanations, reasoning logic, and principles related to the task. The *Examples* select corresponding input-output pair samples from a samples library according to the entity and field, which ensures that ICL samples that meet the current task can be provided for LLMs. The *Input* part contains input information in the same format as the examples in Examples, divided into two parts: context and question. The former is obtained through Hybrid RAG technology, while the latter

Instruction_Counter

You are a legal assistant who needs to ... and accurately answer the number of instances .... Please note the following:

(1) The number of instances refers to ... (**concept explanation**);

(2) Please strictly control the output to be a simple integer, so that .... Note that the range of values for all quantities is [0,5] (**output constraint**);

(3) Here provides ... in the 'Text' section and t... in the 'Questions' section. You should first ..., and then .... Finally, ...(**CoT**).

(4) Finally, please note that .... If there is no content about the agreement in the text, please answer 0 (**some principles and encouragements**).

Examples

{EXAMPLES} (**ICL examples from lib**)

Input

Please practice below:

Text: {CONTEXT} (**RAG Retrieved**)

Question: {QUESTION} (**Dynamically constructed based on entity and field**)

Answer:

Output

{NUMBER} {**an integer as output**}

Figure 3: TST prompt template for instance counter.

Instruction_Extractor

You are a legal assistant who needs to read a judgment document of a civil lending case and ...(**background information**)... following:

(1) Here provides the content of the judgment document in the 'Text' and specific requirements in the 'Questions' section. You should first ..., and then ..., without... (**CoT**).

(2) Please note that when valid agreement appears in the question,.....(**some principles**).

(3) The text may not contain any valid information, so do not blindly follow it. If ....., please boldly answer **<NOT FOUND>**.

Examples

{EXAMPLES} (**ICL examples from lib**)

Input

Please practice below:

Text: {CONTEXT} (**RAG Retrieved**)

Question: {QUESTION} (**Dynamically constructed based on entity and field**)

Answer:

Output

{STRING} {**can constrcut triplet with entity and field**}

Figure 4: TST prompt template for instance extractor.

generates grammatically correct questions based on rules. The final *Output* part is the output with specific format requirements. Sometimes, LLMs cannot be output according to format requirements. Solutions include breaking down complex output formats into multiple outputs or building a format fine-tuning dataset to fine-tune LLMs.

C Experiment Setting

As shown in Table 2, we test all methods on different datasets and metrics.

D Dynamic Part of CPL Dataset

For CPL dataset, we select the dynamic fields as a sub-dataset, which consists of *names* of the plaintiff, defendant, and court, as well as the **six main fields** of the court and the plaintiff regarding the *lending voucher*, *agreed lending amount*, *agreed repayment date*, *agreed interest*, *agreed overdue interest*, and *agreed liquidated damages*, and **dozens of sub-fields** such as the *name and content of the lending voucher*, *the agreed form*, and *agreed time*. It has difficulties such as multi-perspective interference, multi-instance interference, and inter-field dependencies. Fig. 5 shows the results of distribution statistics on the above-mentioned subjects and fields.

Like LiveSum, we also classify the difficulty of different fields into three levels: green represents Easy, blue represents Medium, and red represents Hard. It can be seen that there are some patterns in the distribution of fields, among which, in the simple name field, the court has uniqueness, the vast majority of plaintiffs are unique, and there can be multiple defendants. In the blue field, most instances are within two instances, with one instance being the majority case. In the red fields, the majority are zero, and some fields have less than one-tenth of their non-zero instances, which is a standard tail field. Therefore, simple fields

Datasets	Methods	Models	Type	Metrics
LiveSum	T^3	Previous works	/	RMSE | ER
	T^3	Mistral-7B-Instruct-v0.2	Original	RMSE | ER
	Pure Rule	/	/	RMSE | ER
	TST	Llama-2-7B-Chat	Original	RMSE | ER
	TST	Llama-3-8B-Instruct	Original | FT	RMSE | ER
	TST	Baichuan2-7B-Chat	Original	RMSE | ER
	TST	ChatGLM3-6B	Original | FT	RMSE | ER
	TST	Mistral-7B-Instruct-v0.2	Original | FT	RMSE | ER
	TST	Qwen1.5-7B-Chat	Original | FT	RMSE | ER
	TST	Qwen2.5-0.5B	Original | FT	RMSE | ER
CPL (D)	Naive	GPT-4o with JSON Schema	Original	RMSE | ER
	All Zero	/	/	RMSE | ER
	All One	/	/	RMSE | ER
	T^3	GPT-4	Original	RMSE | ER | F1-Score
	T^3 | TST | TKGT	ChatGLM3-6B	Original | FT	RMSE | ER | F1-Score
	T^3 | TST | TKGT	Mistral-7B-Instruct (zh)	Original | FT	RMSE | ER | F1-Score
	T^3 | TST | TKGT	Qwen1.5-7B-Chat	Original | FT	RMSE | ER | F1-Score
	T^3 | TST | TKGT	Baichuan2-7B-Chat	Original	RMSE | ER | F1-Score
	T^3 | TST | TKGT	Qwen2.5-0.5B	Original	RMSE | ER | F1-Score

Table 2: Experiment settings of the second stage of TST. Experiments on E2E, Rotowire, CPL (S) are designed to demonstrate the performance of TKGT, while experiments on LiveSum and CPL (D) are for TST.

are mainly responsible for examining the ability of extraction methods to extract accurately, medium fields focus on counting and distinguishing, and hard fields challenge the model's ability to say *I do not know*.

E Results on CPL with the Triplet Metric

As shown in Table 3, we test all methods on the CPL dynamic dataset with the triplet metric. The analysis content can be found in the experiment part.

F Results on CPL with the Counting Metric

As shown in Table 4, the TST method will first generate statistics about the number of times, but other methods directly generate results, so we deduced the count from the results and obtained Table 4. Similarly, we have added two outcome baselines, all zeros, and all ones, besides the GPT-4o baseline. It can be seen that GPT-4o (Json Schema) achieved a very low RMSE of 0.006 and ER of 0.47 on the Easy field, which exceeds most subsequent methods. However, GPT-4o performs poorly in the Medium and Hard fields, even far below all ones and all zeros. Through sample analysis, we found that GPT-4o is very negative in answering questions that are not known, which is also the reason for its poor performance in the tail fields. In addition, in the comparison of T^3, TKGT, and TST methods, we can clearly see the significant advantage of TST in counting. Significantly, the Qwen1.5-7B Chat and Chinese-Mistral-7B Instruction models outperform other methods in all fields after being fine-tuned. Therefore, a significant portion of the superiority of TST comes from its first step of counting the number of instances. This information about the number of instances can effectively improve the accuracy of information extraction results.

Methods	The first column F1			Table header Fl			Data cell F1		
	Exact	Chrf	BERT	Exact	Chrf	BERT	Exact	Chrf	BERT
GPT-4o (Json Schema)	100.00	100.00	100.00	69.69	78.10	82.50	57.63	62.90	66.83
T^3									
ChatGLM3-6B	2.16	3.55	3.42	0.50	1.02	1.29	0.19	0.25	0.25
Qwen1.5-7B-Chat	31.10	31.50	31.80	0.92	5.97	8.74	0.99	1.67	1.91
Mistral-7B-Instruct (zh)	0.24	0.24	0.24	0.00	0.09	0.13	0.00	0.00	0.01
Baichuan2-7B-Chat	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Qwen2.5-0.5B	0.19	0.19	0.19	0.00	0.00	0.00	0.00	0.00	0.00
ChatGLM3-6B (FT)	83.70	85.05	86.11	55.11	62.11	66.39	42.82	**47.16**	**50.01**
Qwen1.5-7B-Chat (FT)	97.77	97.91	98.01	35.00	36.77	38.38	**43.13**	43.96	44.51
Mistral-7B-Instruct (zh, FT)	59.15	59.15	59.15	21.44	22.32	23.58	27.22	27.44	27.57
GPT-4	**98.89**	**98.90**	**98.91**	**60.77**	**70.37**	**75.92**	30.48	35.17	36.51
TKGT									
ChatGLM3-6B	93.93	94.72	95.27	36.77	54.72	64.39	1.81	6.57	8.71
Qwen1.5-7B-Chat	**100.00**	**100.00**	**100.00**	36.12	55.81	66.10	15.22	28.51	34.68
Mistral-7B-Instruct (zh)	98.01	98.29	98.49	36.22	55.32	65.43	5.00	19.95	23.23
Baichuan2-7B-Chat	74.74	77.27	79.11	23.92	37.54	47.46	0.50	2.05	2.93
Qwen2.5-0.5B	**100.00**	**100.00**	**100.00**	35.20	54.19	64.43	0.78	8.14	10.96
ChatGLM3-6B (FT)	**100.00**	**100.00**	**100.00**	51.54	66.77	74.66	37.19	45.05	50.12
Qwen1.5-7B-Chat (FT)	**100.00**	**100.00**	**100.00**	73.65	80.74	84.65	66.14	70.53	73.57
Mistral-7B-Instruct (zh, FT)	**100.00**	**100.00**	**100.00**	**77.09**	**83.08**	**86.41**	**67.85**	**72.02**	**74.93**
TST									
ChatGLM3-6B	97.16	97.26	97.43	69.50	75.55	79.11	65.40	67.55	69.32
Qwen1.5-7B-Chat	99.72	99.76	99.78	84.18	88.07	90.20	83.56	85.15	86.41
Baichuan2-7B-Chat	98.34	98.47	98.57	**97.66**	**97.93**	**98.11**	0.65	0.71	0.77
Mistral-7B-Instruct (zh)	99.81	99.84	99.86	72.26	79.90	83.67	73.04	74.70	76.02
Qwen2.5-0.5B	97.91	98.09	98.25	66.96	74.82	79.01	64.86	66.17	67.26
ChatGLM3-6B (FT)	**100.00**	**100.00**	**100.00**	86.23	89.71	91.54	85.98	87.53	88.83
Qwen1.5-7B-Chat (FT)	**100.00**	**100.00**	**100.00**	86.93	90.83	92.24	86.58	88.07	89.32
Mistral-7B-Instruct (zh, FT)	**100.00**	**100.00**	**100.00**	87.65	90.83	92.55	**87.15**	**88.65**	**89.87**

Table 3: Results of T^3, TKGT, and TST methods on CPL datasets using the triplet metric. *Json Schema* is an API function provided by advanced models such as GPT-4o, which can force the output of LLMs to be structured. Besides, adding the letters *FT* after the model indicates that it's a fine-tuned model, and *zh* means its a Chinese fine-tuned version.

Model	Easy		Medium		Hard		Average	
	RMSE	ER	RMSE	ER	RMSE	ER	RMSE	ER
GPT-4o (Json Schema)	**0.006**	**0.47**	1.032	52.29	1.489	88.78	1.185	56.52
All Zero	0.891	100.00	**0.258**	**19.75**	0.728	68.96	0.665	55.48
All One	0.240	15.17	0.633	81.99	**0.450**	**44.87**	**0.541**	**53.78**
T^3								
ChatGLM3-6B	0.888	98.89	0.277	20.46	0.766	69.12	0.691	55.61
Qwen1.5-7B-Chat	0.859	95.42	0.428	26.62	0.742	69.04	0.724	57.35
Baichuan2-7B-Chat	0.891	100.00	**0.258**	19.75	**0.728**	68.95	0.665	55.48
Mistral-7B-Instruct (zh)	0.891	100.00	0.260	19.98	**0.728**	69.04	0.665	55.61
Qwen2.5-0.5B	0.891	100.00	**0.258**	19.75	**0.728**	68.96	0.665	55.48
ChatGLM3-6B (FT)	0.549	57.35	0.304	**19.04**	0.900	67.93	0.702	46.26
Qwen1.5-7B-Chat (FT)	0.085	3.79	0.289	20.22	0.765	68.33	**0.555**	**36.18**
Mistral-7B-Instruct (zh, FT)	0.370	41.07	**0.258**	19.75	0.734	**67.14**	0.576	42.97
GPT-4	**0.030**	**3.0**	0.881	36.57	1.195	77.65	0.973	46.29
TKGT								
ChatGLM3-6B	1.561	91.94	0.412	27.65	2.026	79.78	1.581	61.36
Qwen1.5-7B-Chat	**0.021**	**1.58**	2.283	59.40	2.370	91.07	2.260	60.51
Mistral-7B-Instruct (zh)	0.086	5.69	1.186	59.32	2.875	94.08	2.032	62.50
Baichuan2-7B-Chat	1.058	100.00	0.361	24.49	1.188	72.59	0.976	58.83
Qwen2.5-0.5B	1.098	87.20	0.901	50.00	2.542	89.10	1.847	73.08
ChatGLM3-6B (FT)	0.256	16.59	0.968	51.42	2.379	91.39	1.704	60.44
Qwen1.5-7B-Chat (FT)	0.240	15.17	0.285	22.04	0.941	64.30	0.664	37.57
Mistral-7B-Instruct (zh, FT)	0.240	15.17	**0.255**	**19.67**	**0.939**	**58.37**	**0.658**	**34.25**
TST								
ChatGLM3-6B	0.444	26.54	0.419	34.20	0.643	57.74	0.588	42.09
Qwen1.5-7B-Chat	0.056	4.74	0.270	21.25	0.390	31.12	0.355	21.90
Mistral-7B-Instruct (zh)	0.025	2.05	0.497	48.10	0.593	46.52	0.532	38.26
Baichuan2-7B-Chat	0.330	26.22	0.276	22.12	0.583	46.05	0.483	32.51
Qwen2.5-0.5B	0.533	48.82	0.576	37.68	0.804	70.93	0.733	53.21
ChatGLM3-6B (FT)	0.209	13.74	0.255	20.54	0.413	38.23	0.376	26.26
Qwen1.5-7B-Chat (FT)	0.010	0.79	0.149	11.53	**0.314**	**25.67**	**0.256**	15.04
Mistral-7B-Instruct (zh, FT)	**0.002**	**0.16**	**0.140**	**10.82**	0.332	25.91	0.260	**14.72**

Table 4: Results of T^3, TKGT, and TST methods on CPL datasets using the counting metric. *Json Schema* is an API function provided by advanced models such as GPT-4o, which can force the output of LLMs to be structured. Besides, adding the letters *FT* after the model indicates that it's a fine-tuned model, and *zh* means its a Chinese fine-tuned version.

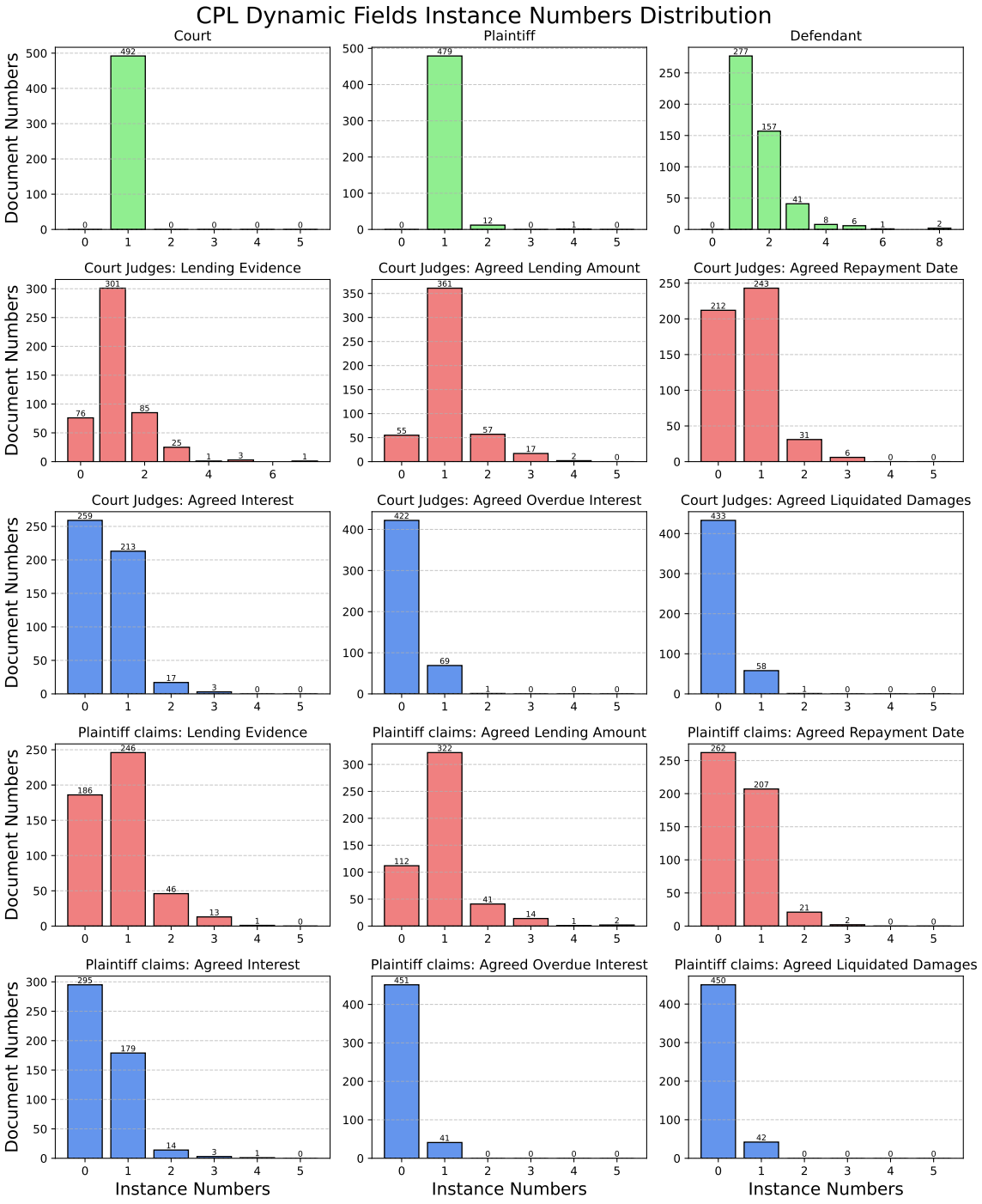

Figure 5: The instance numbers distribution of CPL (D) fields. Colors represent fields' difficulty level: green means easy, blue means medium, and red means hard.