Unstructured Data Fusion for Schema and Data Extraction

KAIWEN CHEN, University Of Toronto, Canada
NICK KOUDAS, University Of Toronto, Canada

Recently, there has been significant interest in extracting actionable insights from the abundance of unstructured textual data. In this paper, we introduce a novel problem, which we term Semistructured Schema and Data Extraction (*SDE*). This task aims to enhance and complete tables using information discovered from textual repositories, given partial table specifications in the form of queries. To effectively solve *SDE*, several challenges must be overcome, which involve transforming the partial table specifications into effective queries, retrieving relevant documents, discerning values for partially specified attributes, inferring additional attributes, and constructing an enriched output table while mitigating the influence of false positives from the retrieval.

We propose an end-to-end pipeline for *SDE*, which consists of a retrieval component and an augmentation component, to address each of the challenges. In the retrieval component, we serialize the partial table specifications into a query and employ a dense passage retrieval algorithm to extract the top-k relevant results from the text repository. Subsequently, the augmentation component ingests the output documents from the retrieval phase and generates an enriched table. We formulate this table enrichment task as a unique sequence-to-sequence task, distinct from traditional approaches, as it operates on multiple documents during generation. Utilizing an interpolation mechanism on the encoder output, our model maintains a nearly constant context length while automatically prioritizing the importance of documents during the generation.

Due to the novelty of *SDE*, we establish a validation methodology, adapting and expanding existing benchmarks with the use of powerful large language models. Our extensive experiments show that our method achieves high accuracy in enriching query tables through multi-document fusion, while also surpassing baseline methods in both accuracy and computational efficiency.

CCS Concepts: • **Information systems** → *Information extraction*; • **Computing methodologies** → **Information extraction**.

Additional Key Words and Phrases: Data Fusion, Information Extraction, Schema Extraction

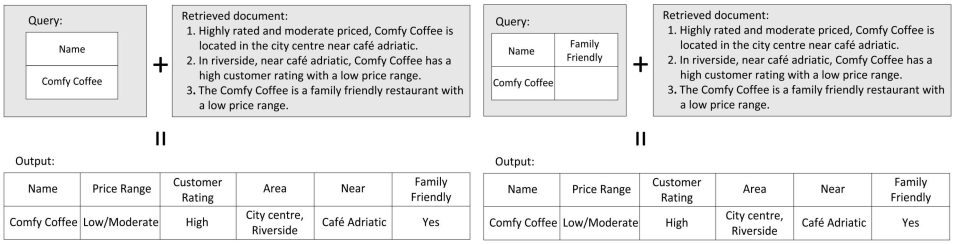
ACM Reference Format:
Kaiwen Chen and Nick Koudas. 2024. Unstructured Data Fusion for Schema and Data Extraction. *Proc. ACM Manag. Data* 2, 3 (SIGMOD), Article 181 (June 2024), 26 pages. https://doi.org/10.1145/3654984

1 INTRODUCTION

Unstructured data, predominantly in textual format, has become incredibly common. Over time, the fields of data management and related communities have undertaken focused research efforts aimed at extracting valuable insights from textual data. This has led to the development of techniques like entity and relationship extraction, as well as table extraction [10, 11, 24, 26, 37, 48]. These techniques have witnessed substantial progress in recent years, thanks to the emergence of Large Language Models (LLMs) [34, 44, 46]. These models have significantly improved our ability to capture semantics and context effectively[5, 28, 32].

Authors' addresses: Kaiwen Chen, University Of Toronto, Toronto, Ontario, Canada, kckevinchen@cs.toronto.edu; Nick Koudas, University Of Toronto, Toronto, Ontario, Canada, koudas@cs.toronto.edu.

ACM 2836-6573/2024/6-ART181
https://doi.org/10.1145/3654984



(a) A fully specified attribute (b) A partially specified attribute

Fig. 1. Two SDE example of venue's comments: a) A fully specified attribute. The inquiry includes the name of the venue. b) A partially specified attribute, "Family Friendly", is introduced, prompting the system to provide a value if feasible.

Recent years have witnessed paramount research focus within the data management community on the topic of data discovery [3]. While the terminology may appear general in nature, the majority of this research endeavors to leverage extensive collections of structured data, potentially augmented with external data sources, to achieve several distinct objectives[12, 14–16]. These objectives encompass the identification of data sets closely associated with existing ones, the discernment of tables amenable to joining operations, and the execution of union table searches, all aimed at enhancing the overall utility and comprehensiveness of existing data repositories [13, 16].

In this paper, we introduce a novel problem, which we call *Semistructured Schema and Data Extraction* (*SDE*). Our main goal is, given *partial* table specifications as queries, to *enhance* (by adding more related attributes) as well as *complete* (by populating the enhanced table representation with additional rows as applicable) the tables with information discovered from textual repositories. To realize things concrete, consider the example depicted in Figure 1. Given a large collection of comments provided by users for venues, we would like to automatically extract descriptive information for a specific venue. We start by issuing a query (on the suitably indexed collection of comments via traditional [1, 40] or neural index [10, 20, 30, 36] or both) with the name of the venue we are interested in, expressed as a table (in Figure 1a) having an attribute 'Name' and a target attribute value 'Comfy Cafe'. In this case, we will say that the attribute 'Name' is fully specified since we provide a value for it for our search. We collect all comments containing the venue name (the attribute value) in some ranked order in return. Alternatively, we may include an additional attribute in our search, such as 'Family Friendly' without specifying a value for it. In this case we will say that attribute 'Family Friendly' is partially specified (Figure 1b). That way, we indicate that the comments retrieved should contain information about how family-friendly the venue is, which we wish to process further. We will fully detail how such partial table specifications are converted into information retrieval queries in Section 3.1.2.

The desired output, after retrieving and processing suitable information, has a tabular format. The table specification in the output includes all fully specified attributes along with their specified values, all partially specified attributes along with *values inferred* for them as a result of processing the retrieved information and *additional* applicable attributes along with their values. Notice that the total number of attributes in each case depends on the initial query table specification as the documents retrieved are query dependent.

For further illustration, let us contemplate a scenario involving a corpus of news documents centred around sports, as depicted in Figure 2. In this particular context, the query delineates the extraction of two distinct tables: one encompassing the names of the target teams and the other

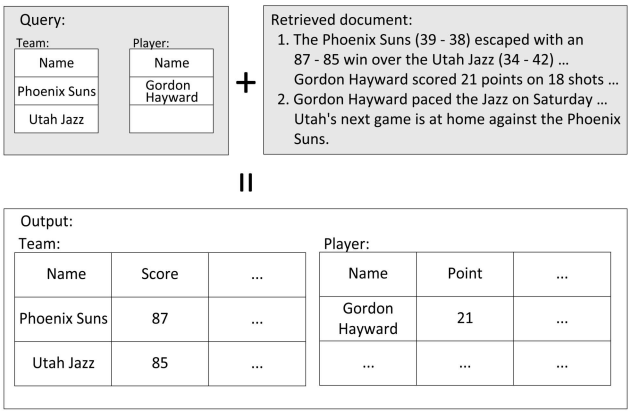


Fig. 2. An SDE example from basketball game summaries. The query comprises multiple records (teams and players). The retrieved summary should pertain to both the team names and player names mentioned in the query.

enumerating the names of specific players. The overarching objective here is to augment each of these tables with contextually relevant attributes sourced from news articles that chronicle games between the aforementioned teams and provide detailed insights into the performance of the specified players within these game contexts. The resultant outcome of this process manifests as two enriched tables, imbued with supplementary attributes and corresponding values extracted from the retrieved articles, thereby enhancing the utility of the team and player name datasets.

Two unique features of *SDE* set it apart from existing information extraction problems studied in the literature. First, in *SDE*, the extraction and integration process is driven entirely by the data itself, adapting to its structure and content without relying on *a predefined schema*. This approach contrasts with traditional Knowledge graph construction algorithms [19, 50] or more specifically, entity and relation extraction [49, 53], where the extraction processes often rely on a predefined schema. Essentially the schema extracted in *SDE* is not set a priori but is adaptive to data in the same domain as more data is encountered. This allows for greater flexibility to accommodate diverse data types and to derive highly complex table structures. As evidenced by prior work[48], the flexibility in the derived schema and the complexity of the output structure make traditional entity and relation extraction less competitive even when a carefully designed schema is provided. Additionally, SDE leverages *partial table specifications* as queries, guiding the extraction process in a more targeted manner. This introduces new challenges during data discovery, where the model is required to automatically identify relevant documents pertaining to the queries.

Second, *SDE* does not make any assumptions on the quality or the number of input documents. This also includes documents with potentially conflicting information that need to be handled by the model. The enriched table in *SDE* is designed to contain all potential values in the case of conflicts. By including all potential values, *SDE* allows for a more nuanced analysis and aids downstream applications to understand and resolve these conflicts. Such flexibility enables *SDE* to handle a wide variety of data sources, making it highly adaptable for different application scenarios. One motivating application for *SDE* comes from the domain of clinical trails and medical systematic reviews [29, 45]. Researchers need to aggregate and synthesize information from multiple reviews to enhance the comprehensiveness and reliability of their analysis. This is laborious manual process and automation (as enabled by *SDE*) can make the process more efficient. Of equal importance, in

enterprise applications, synthesizing information and reports from multiple sources is a typical task, e.g., conducting research in the automotive industry to synthesize data from various sources (web, reviews, trade publications, etc) in order to create a comprehensive overview of electric vehicles (EV) trends (models, reviews, prices, technical specs taking into account multiple possibly conflicting sources for the same model for example). Similar types of aggregate information views across numerous related documents abound in industry verticals (e.g., electronics, consumer products etc). Moreover, while building machine learning systems [17], researchers have observed that model performance often scales with the number of data sources. Researchers on table discovery [7, 52] have successfully created and augmented high-quality relational training datasets from structured data lakes to boost the performance of machine learning models. *SDE* complements these augmentation efforts with extractions from unstructured data, allowing researchers to enrich machine learning models with a broader spectrum of information. In short, the ability to support multi-source data integration with a flexible schema becomes an essential distinction and challenge for *SDE* compared to prior work [48].

To address this challenge comprehensively, several key hurdles must be surmounted. Firstly, we must undertake the transformation of partial table specifications into queries that are tailored for downstream utilization in document retrieval. Secondly, once these queries are established, it becomes imperative to retrieve all pertinent documents. Thirdly, an automated mechanism must be implemented to discern appropriate values for any attributes that remain partially specified within the query. Fourthly, we need to engage in the inference of a suitable set of supplementary attributes, complete with their respective values, as deemed relevant, and subsequently assemble an output table. Lastly, it is crucial to acknowledge the potential presence of false positives within the retrieved documents, as exemplified in Figure 2, where Document 2, in the context of the query, contributes insignificantly to the construction of the output table and thus qualifies as a false positive concerning the *SDE* problem. Our proposed technique should possess the capability to identify and mitigate the impact of such results on the final record's construction.

We present an end-to-end solution to this problem. After generating a suitable query given a (partial) table input specification (Section 3.1) we utilize off-the-shelf information retrieval or neural retrieval systems [20] to retrieve all relevant documents. To assemble the output record, we design a model to jointly process the user query and the candidate retrieved documents, casting the problem as a sequence-to-sequence problem. The objective of the model is to capture suitable content from all retrieved documents. This is different from traditional sequence-to-sequence [2, 43, 48] tasks in which the objective is to operate on a single input (document). Thus, our proposal handles multiple documents during the generation of the output record while carefully extracting only relevant information from each input document. In order to accomplish this, we introduce an *interpolation* mechanism that effectively prioritizes the importance of each document for generation. More importantly, this mechanism is implemented on the encoder output, enabling the model to accept multiple documents as input without increasing the length of the encoder output, thereby maintaining the size of the model. We learn this interpolation and utilize a constructive learning paradigm in a supervised manner [22].

As *SDE* is a novel problem, we have to devise a validation methodology from first principles. Although prior work in the data extraction area focuses on extracting tables from a single document [26, 48], the evaluation methodologies utilized in the past are not applicable in our setting. Thanks to the recent advances in large language models (LLMs) with superior performance in language understanding and generation[34, 44], we could adapt and expand benchmark datasets for *SDE*. In these benchmarks, the information pertinent to an output record span could potentially multiple documents. These benchmarks are now available to the community as part of our work, providing valuable resources for further research

We present a set of extensive experiments to verify our design using real and synthetic data. Results show that our method is highly accurate when fusing information from multiple documents to generate representative output records. Moreover, when compared to baseline methods, our techniques not only exhibit higher accuracy but do so at a fraction of the time required by other applicable approaches, thus exhibiting superior performance.

In summary, in this paper, we make the following contributions.

- We introduce the *SDE* problem, a novel problem in the domain of automated data extraction. The technique is straighforward to utilize with traditional information retrieval [40] or neural retrieval systems [20] or both.
- We present an end-to-end solution to the *SDE* problem consisting of a novel model architecture and an interpolation technique enabling the processing of multiple documents after information retrieval with the ability to effectively de-noise irrelevant information for improved accuracy. We refer to our proposal as Table Enrichment with Interpolation (**TEI**).
- We introduce a methodology to adapt existing benchmark data sets (for validation purposes) to *SDE*, and we make these data sets available to the community for future work.
- We present a comprehensive set of experiments using both real and synthetic data sets, demonstrating that our end-to-end solution to the *SDE* problem offers superior accuracy and performance when compared to other applicable approaches.

2 PROBLEM FORMULATION

The objective of *SDE* is to enhance query tables by sourcing information from a collection of documents. The system ingests tables as input, selects pertinent documents from the collection, and then generates enriched tables that expand upon the original, chiefly by adding additional attributes and associated values with data derived from the documents. To keep our presentation simple, we assume that the fully specified attributes in the query table are unambiguous (e.g., in Figure 2 the value 'Phoenix Suns' uniquely corresponds to a basketball team). If that is not the case, entity disambiguation techniques can be applied to filter the retrieved document set.

Formally, consider a collection of documents denoted as D. The query for *SDE* is a set of tables $T = \{t_1 \dots t_n\}$. Within this set, a table t_k is assumed to contain $n_{k,r}$ rows and $n_{k,c}$ columns. The value associated with a specific row i and column j is expressed as $c_{k,i,j}$. We assume the first row of each table t_k, $c_{k,1,j}$, contains the attribute names (i.e. Name, Price Range) and the subsequent rows $c_{k,i,j}, i > 1$ hold the values corresponding to these attributes. We assume that the attribute names, $c_{k,1,j}$ are non empty. However, the values for some, but not all, of those attributes can be partially specified, namely $c_{k,i,j}, i > 1$ may be empty.

Assume that a query Q is derived from T and issued against D. The result of this query is a subset of documents from D, denoted as $X = \{d_1 \dots d_m\}$. The output of the *SDE* is a *set of enriched tables* $T' = \{t'_1 \dots t'_n\}$ such that $|T| = |T'|$.

Each enriched table $t'_k \in T'$ encompasses the information of t_k and extends it utilizing the information in X introducing new attributes (columns) along with their corresponding values or by providing a value for each partially specified attribute of t_k. The information contained in the original T is preserved such that for the table $t_k \in T$ that corresponds to t'_k, all attributes $c_{k,1,j}$ in t_k should also exist in t'_k, denoted as $c_{k',1,j}$. Additionally, for all the value pair $c_{k,i,j} \in t_k, c_{k',i,j} \in t'_k, i > 1$ pertaining to the attribute, $c_{k',i,j}$ is a super-set of $c_{k,i,j}$.

3 SYSTEM STRUCTURE AND RETRIEVAL MODEL

In this section, we present the details of our proposed solution designed for the *SDE* problem. As illustrated in figure 3, the proposal consists of two components: retrieval and augmentation.

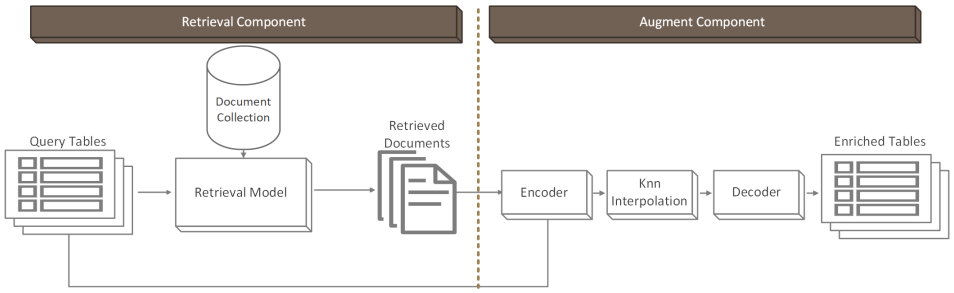


Fig. 3. An overview of the system's architecture. It mainly consists of two modules: The first module is the retrieval component, which extracts documents relevant to the query table from the document collection; the second module is the augment component, which takes the retrieved documents and the query tables as input and outputs an enriched version of each query table.

3.1 Retrieval Component

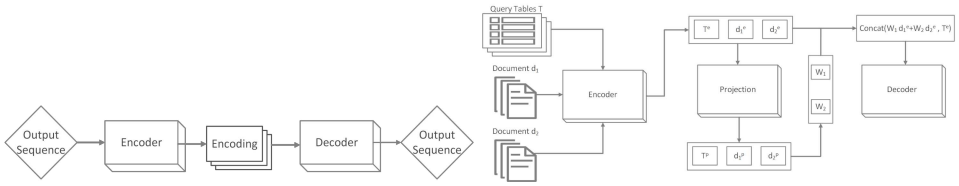
The retrieval component addresses the challenge of document retrieval from a given unstructured data collection. When presented with a query table T and a collection of documents D, the retrieval component is designed to output the top k documents $d_1, d_2, \ldots, d_k$ that best match the query. We adopt standard techniques for this [20] and emphasize the flexibility of the retrieval component, highlighting that the choice of the retrieval techniques is orthogonal to the overall system architecture. This enables seamless integration of diverse retrieval methods without affecting the primary design.

3.1.1 ***Retrieval Model.*** Without loss of generality in this work, we have chosen to leverage the *DPR* (Dense Passage Retrieval) model, which is currently recognized as one of the state-of-the-art passage retrieval methods [20]. We selected DPR due to its exceptional performance in terms of accuracy when compared to other retrieval methods like tf-idf and BM25 [40, 41]. The *DPR* model incorporates a bi-encoder architecture that separately encodes the query and the text from and each document of the collection into d-dimensional vectors using neural embedding techniques [10, 42]. The measure of similarity between the query and a target document is quantitatively determined through the dot product of their corresponding vectors. This approach provides a nuanced, yet efficient, representation of relevance, thereby optimizing the retrieval of pertinent documents from the collection documents.

As our input query T consists of structured data, we need first to serialize input into a textual format that is congruent with the specifications of the *DPR* model.

3.1.2 ***Query Serialization.*** We incorporate two special tokens designed to differentiate between rows and columns within a singular table. Specifically, the row separation token $< R >$ and the column separation token $< C >$ are used to partition row cells and column cells within a table respectively. Following the notation of section 2, given a single table t with n_r rows and n_c columns, the serialization is defined as

$$
\begin{aligned}
Serialized(T) = & < R > c_{1,1} < R > & \ldots < R > c_{n_r,1} < R >< C > \\
& < R > c_{1,2} < R > & \ldots < R > c_{n_r,2} < R >< C > \\
& & \ldots\ldots \\
& < R > c_{1,n_c} < R > & \ldots < R > c_{n_r,n_c} < R >
\end{aligned}
$$



(a) Encoder-Decoder (b) Our Model **TEI**

Fig. 4. The Data Flow for the original Encoder-Decoder based model (a) and our proposed model **TEI**(b).

For the query $T = \{t_1 \dots t_n\}$ with multiple tables, an additional separation token $< T >$ is used such that:

$$Serialized(Q) = Serialized(t_1) < T > \cdots < T > Serialized(t_n)$$

Furthermore, in cases where the table names are present, they will be encoded as a prefix to each of the table serializations $Serialized(T)$.

To ease notation, we denote a serialized table as T^s in what follows. Although we chose *DPR* as the retrieval component in this work, it should be reemphasized that our framework can adapt to any retrieval method with minimal modifications. For example, to accommodate more traditional retrieval methods [40], the only required changes would be to serialize the query tables by omitting the attribute description.

4 TABLE ENRICHMENT WITH INTERPOLATION

In this section, we introduce our solution to *SDE*, Table Enrichment with Interpolation (**TEI**). We conceptualize SDE as a sequence-to-sequence problem. We propose and develop a number of extensions to adapt the basic architecture to the SDE problem specifics.

4.1 Encoder-decoder Model

The model's architectural framework is rooted in the encoder-decoder paradigm[25, 37, 39, 43], as illustrated in Figure 4a. In the context of sequence-to-sequence generation, the encoder module assumes the role of converting the input sequence into a high-dimensional encoding vector, effectively compressing the input for subsequent processing. Conversely, the decoder component orchestrates the synthesis of the output sequence under the guidance of this encoding vector.

Our proposed solution strategically tackles the architectural challenges arising from the linear expansion of the input sequence concerning the number of retrieved documents. Furthermore, it addresses the nuanced issue of document relevance, acknowledging that certain documents bear greater significance to the task at hand, while also adeptly managing the presence of noisy or irrelevant documents.

To address these concerns, we introduce an innovative K-Nearest Neighbors (KNN) interpolation technique, which amalgamates all input sources into a cohesive encoding representation while effectively mitigating the impact of irrelevant documents. Subsequently, the unified encoding is harnessed by the decoder to facilitate the generation of the output sequence.

4.2 Knn Interpolation

Given a serialized query T^s along with a set of k documents $\{d_1 \dots d_k\}$ from the output of the retrieval component, as shown in figure 4b, the k-nearest-neighbour interpolation is effectively analogous to a form of attention built on top of individual documents. Intuitively, the interpolation

mechanism guides the model to focus more on the informative documents while filtering out potential noise present in the retrieved k-document set.

For this purpose, we introduce two linear projection layers $proj_T$ and $proj_k$, tailored for the query and the documents, respectively. The intent behind these projections is to transform the encoder output into more effective representational spaces for both the query and documents. Let T^e and d_i^e denote the encoder output of the serialized query and the i-th document, respectively.(i.e. $T^e = encoder(T^s)$ and $d_i^e = encoder(d_i)$. The interpolation weight w_i is computed as follows:

$$
\begin{aligned}
T^p &= proj_t(T^e) \\
d_i^p &= proj_k(d_i^e) \\
w_i &= Softmax((T^p \cdot d_i^p + \sum_{i \neq j} d_i^p \cdot d_j^p))
\end{aligned}
$$

where the softmax is taken with respect to all the documents.

It is worth noting that during the computation of the weight assigned to a document, the mechanism employs more than just the dot product with the query T. In fact, all other documents within the retrieval are also taken into account. This design choice is predicated on the intuition that noise-inducing documents are frequently dissimilar to other documents within the same retrieval set.

After computing the interpolation weight w_i for each of the k documents, the output of the k-nearest-neighbour interpolation is a weighted average of all documents concatenation with the query.

$$Concat(\sum w_i * d_i^e, T^e)$$

Subsequently, the decoder utilizes the computed interpolation as its input to generate the final output sequence. During this interpolation phase, we obviate the necessity of linearly increasing the token length in proportion to the number of documents, a requirement when naively concatenating all documents. Instead, we set the token length to be the sum of the maximum document token length and the query token length, which enhances the scalability of the model.

An alternative design choice, could simply add all the encodings for the retrieved documents obtaining a single vector. This vector would then be provided to the decoder to generate the enriched table. This design addresses the issue of linearly increasing token length, where the required token length remains constant with respect to the number of retrieved documents. However, this approach doesn't account for any potential noise or for the relative importance of the information within the retrieved documents to the query. Therefore, the end performance may be affected, especially when noise is present in the retrieved documents. We will explore and compare this alternative design with our proposed method in an upcoming ablation study.

4.3 Model Training

We train each component independently, allowing us to seamlessly integrate any existing retrieval method from data discovery without incurring extra costs.

4.3.1 ***Retrieval Training.*** For the retrieval component, the *DPR* training framework is utilized [20]. Within this framework, the model is presented with both positive and negative labelled documents related to a specific input query. The learning objective is to construct a vector space in which relevant query-document pairs are closer in terms of distance while concurrently maximizing the distance between irrelevant pairs. It is achieved by optimizing the negative log-likelihood associated with the positively labelled documents.

4.3.2 ***Augmentation Training.*** We denote a training batch as $B = \{S_1 \dots S_n\}$. A single instance S in the batch is a triplet consisting of a serialized query table T^s, its associated relevant documents $\{t_1 \dots t_m\}$ and the ground-truth serialized enriched table T^s_{target}. To simplify the notation, we represent T^s as t_0. Hence, the input to the model can be compactly described as $P = \{t_0, t_1 \dots t_m\}$. Additionally, the model output corresponding to the input P is denoted as $T^s_{predict}$.

To train the augmentation component, we first employ conventional methods commonly used in sequence-to-sequence frameworks [48]. Specifically, we define a sequence-to-sequence loss

$$L_s = CE(T^s_{predict}, T^s_{target})$$

which is the cross-entropy loss calculated between the predicted and target enriched tables.

Furthermore, to facilitate the Knn interpolation, we design a contrastive regularization that tailors the encoder and projection output to discern the relevance between the query and the documents. This is achieved through a supervised contrastive learning framework [6, 22] and the loss is formulated as follows:

$$L_c = \sum_{P_i \in B} \frac{-1}{|P_i|} \sum_{t'_k, t'_p \in P_i} \log \frac{exp(t'_k \cdot t'_p / \tau)}{\sum_{t'_q \in \cup B} exp(t'_k \cdot t'_q / \tau)}$$

where $t'_k = proj_k(encoder(t_k))$ if $k > 0$ or $proj_t(encoder(t_k))$ if $k = 0$ and τ is the temperature hyper-parameter.

In essence, this contrastive loss encourages the encoder and projection layer to cluster elements within each individual pair while simultaneously maximizing the separation between different pairs.

Consequently, the total training loss for the augmentation component is defined as:

$$L = L_s + \lambda * L_c$$

where λ is a hyper-parameter to balance the effect of the contrastive regularization.

5 EXPERIMENTAL EVALUATION

To validate our method **TEI**, we conduct experiments on three widely used datasets—extended to meet *SDE*'s specific validation requirements—as well as an additional synthetic dataset. We utilize publicly available benchmark data for two reasons. First to assure our results are reproducible and second to make it easier to share the data publicly.

5.1 Datasets

We utilize three existing datasets which are commonly employed in *text-to-table* generation tasks [48]: Rotowire, E2E, and WikiBio. In their original forms, each instance of these datasets features a single document accompanied by one or more tables summarizing their content. However, this format is inadequate for validating the *SDE* methodology, which solves a more general problem and requires data mapping multiple documents to tables. Below, we provide a brief description of each dataset as well as details on our extensions, outlining the added value and improvements.

*SDE-**E2E**.* The E2E dataset [33] includes 50,000 pairs of text and tables, each of which focuses on the restaurant domain. Each instance in the dataset includes text that describes an individual restaurant, and the corresponding table summarizing between 3 to 8 attributes extracted from that text. As the dataset is designed for natural language generation, the entries in the table are selected from a constrained set. Given its focus on natural language generation the entries are selected from a fixed set of options. In total, there are only a few hundred unique restaurant venues.

Dataset	Method	Attribute Header F1			Attribute Values F1			Format Error
		Exact	chrF	rescaled Bert	Exact	chrF	rescaled Bert	
SDE-Wikibio	**TEI**	**77.25**	**83.09**	91.08	**67.84**	**74.39**	**75.88**	0.0
	Text-to-table	76.20	82.30	**91.22**	63.87	73.15	73.95	0.0
	GPT-3.5	56.37	68.38	84.75	47.11	55.72	58.09	0.2
	Llama-2-70B	39.38	52.17	67.23	31.51	38.88	41.49	39.4
SDE-E2E	**TEI**	99.60	99.67	99.84	93.68	94.58	96.86	0.0
	Text-to-table	**99.79**	**99.84**	**99.90**	**95.15**	**95.90**	**97.01**	0.0
	GPT-3.5	94.64	96.47	98.90	81.90	85.78	92.32	0.89
	Llama-2-70B	76.56	88.99	94.62	76.56	79.74	86.92	37.97
SDE-Resume	**TEI**	**99.92**	**99.93**	**100**	**99.13**	**99.74**	**99.67**	0.0
	Text-to-table	NA	NA	NA	NA	NA	NA	NA
	GPT-3.5	98.89	99.00	99.98	93.23	96.20	96.49	0.0
	Llama-2-70B	93.02	94.01	98.46	84.12	88.97	92.82	10.31
SDE-Rotowire-Team	**TEI**	**88.70**	**91.65**	91.16	**85.72**	87.24	**93.07**	0.0
	Text-to-table	88.02	91.09	**94.63**	85.43	**87.32**	92.50	0.0
	GPT-3.5	53.13	63.33	65.48	52.21	55.93	67.25	7.90
	Llama-2-70B	49.38	57.06	57.56	25.18	28.11	57.44	10.33
SDE-Rotowire-Player	**TEI**	**89.15**	**92.27**	94.63	85.08	86.50	92.34	0.0
	Text-to-table	88.60	91.71	**94.87**	**86.62**	**88.00**	**92.59**	0.0
	GPT-3.5	56.06	64.83	72.12	51.59	56.47	67.53	41.44
	Llama-2-70B	54.71	63.54	73.92	22.12	27.00	45.89	69.89

Table 1. Table comparing the performance of our method **TEI** with existing methods across multiple datasets with ground-truth dataset, based on F1 scores for attribute headers and attribute values. NA indicates the model can not generate a result for the dataset, and bold numbers indicate the best performance.

To expand the E2E dataset, we consider synthetically generating restaurant names and map the existing descriptions to generated names. Specifically, we generate 10,000 pseudo restaurant names by querying GPT-3.5-turbo which provides these names. Then, we evenly distribute the existing descriptions among each generated name and create new descriptions by replacing the original restaurant names with the synthetically generated ones. We utilize a simple heuristic to minimize the potential conflict between different descriptions. That is, all the descriptions mapped to the same synthetic name originally pertain to the same restaurant. However, it is still possible that disagreements exist in the dataset. For example, two reviews about the same restaurant could provide totally different scores. In our work, we require the enriched table to include all the potential values for the given attribute. That way conflicting values can be easily observed. This presents a unique challenge present in the E2E dataset where the model needs not only to identify relevant documents pertaining to the query table but also to identify conflicting information and carefully extract all the relevant details.

This results in approximately 10,000 restaurants, with each restaurant having up to 5 descriptions consolidated into a single table summarizing all the descriptions. Each of the descriptions is relatively short and contains only a single sentence. To better validate our proposed method, we

randomly merge some to form larger sentences. This introduces an intentional imbalance in the significance of the documents. Specifically, larger sentences inherently contain richer information than shorter ones, serving as a varying factor for our evaluation.

*SDE-**WikiBio**.* WikiBio is a dataset [23] extracted from Wikipedia biography pages. Each text is extracted from the introduction section of a biography Wikipedia page, while the table originates from the page's infobox that summarizes the page.

Our expansion of this dataset involves splitting the long documents into multiple segments and ensuring that the newly generated segments can be properly summarized by the original table. To do so, we first identify the instances that have relatively long documents. We then employ a Language Model, specifically GPT-3.5-turbo, to partition these documents into multiple parts. We use a carefully designed prompt to ensure that no additional information is introduced and employ few-shot learning, which includes examples that we have constructed manually splitting existing documents, into the prompt, to ensure the quality of the generated parts. This yields more than 10,000 examples, each containing up to three documents along with a summary table.

*SDE-**Rotowire**.* Rotowire is a dataset [47] specializing in sports, with a focus on basketball game reports. Each instance includes a text and two tables: one for team performance and another for individual player performance. The text provides a report on a basketball game, which could potentially include peripheral information, such as seasonal performance metrics for players.

Our extension of the Rotowire dataset follows a similar methodology to our approach with WikiBio. Specifically, we employ GPT-3.5-turbo to split the game reports into multiple, coherent parts. This method generates instances for 5,000 basketball games, with each instance comprising up to three report parts and two summary tables for both team and individual player performance.

*SDE-**Resume**.* In addition, we also create a synthetic dataset consisting of 8000 resumes. Each instance in the dataset features a structured table summarizing the personal information of an individual and job experience, and it is paired with two versions of the resume for each individual. The resumes are generated using predefined templates, with the content populated by a pre-trained generative model [8].

We choose to add this data set as the documents are relatively long compared to other datasets in our study. This allows us to demonstrate scenarios where the input documents exceed the context length of the model.

Note that when enhancing all the datasets, the structured table remains unchanged, while only the corresponding documents are split. Therefore, the structured table is the ground truth and corresponds to the data in the original benchmarks. It is essential to ensure that all content from the table is still present in these split documents. To verify this, we employ entity resolution [49] to check whether the entities listed in each table still appear in the corresponding split documents. Furthermore, if an entity from the table is not found in the generated documents, we prompt the LLM to regenerate these documents until no entity is missing. This approach ensures that the generated documents contain all the necessary information for creating the enriched table. In the ensuing sections, we will refer to the versions of E2E, WikiBio, Rotowire we extend as *SDE*-E2E, *SDE*-WikiBio and *SDE*-Rotowire. Basic statistics for each dataset are presented in Table 2

5.2 Retrieval Component Results

5.2.1 ***Creating Queries**.* Each of the datasets we extend contains data instances that comprise tables and a set of documents corresponding to them. For each data instance, we generate a *query table* corresponding to the instance by randomly removing a subset of attributes (and associated values) from the tables. During this process, we ensure that the query table retains any key attributes

Dataset	Train	Valid	Test
SDE-WikiBio	8000	2000	2000
SDE-E2E	10034	1020	1014
SDE-Rotowire	3398	727	728
SDE-Resume	8000	1000	1196

Table 2. Dataset Statistics

Dataset	Recall@10 (%)	Precision@10 (%)
Rotowire	100	20
Wikibio	100	27
E2E	99.4	49.7
Resume	97.2	19.4

Table 3. Retrieval Recall for Various Datasets.

from the original table, which is typically an attribute called "name" in our setting. As such, by construction, the ground truth documents for each query table would be the documents that pertain to the original tables.

For all our datasets, we create the query tables by randomly removing 90% of the original attributes and associated values. Additionally, we maintain all the corresponding documents for each data instance to form the ground truth document collection related to each query table. During the retrieval process, we use the query tables as the basis for querying and aim to identify the ground truth documents from the collection.

5.2.2 ***Query Result.*** We train a *DPR* model to retrieve the top-k relevant documents from *SDE*-E2E, *SDE*-WikiBio, *SDE*-Rotowire and Resume datasets for each of the query tables.

For each query, we calculate its recall as the ratio of the number of correctly retrieved documents to the total number of documents that are truly relevant, according to the ground truth. Similarly, we calculate precision as the ratio of the number of correctly retrieved documents to the total number of documents retrieved.

The recall and precision metrics (we report Recall@10 and Precision@10) for each dataset are summarized in Table 3. Two observations are immediate: first, the relevant documents are included in the top ten documents in their vast majority. Second, the top documents contain documents which are not relevant (noise or false positives) which affects precision.

5.3 Evaluation Procedure

5.3.1 ***Baseline and Training.*** We conduct experiments in two settings. In the first we assume that the retrieval results are perfect by essentially providing the ground truth documents to various methods evaluated. In this way we wish to evaluate the accuracy of the methods assuming that the input documents are noise free namely do not contain false positives. In the second setting, we feed the results of the retrieval component to the methods and evaluate their accuracy. We refer to the former setting as the *ground truth dataset* and to the later as *the retrieval dataset*.

To the best of our knowledge, no existing method in the literature can be directly applied to solve *SDE*. To compare the efficacy of **TEI** against applicable approaches, we adopt two alternative designs, namely utilizing large language models and an End-to-End Model.

Specifically, for large language models, we utilize Llama-2 [44] series (llama-2-70b-chat-hf) and GPT-3.5-turbo. For each model, we adopt a few-shot learning approach, wherein we use a small, carefully selected set of examples to guide the model. In this setting, we first formulate the template for a single sample:

```
Documents:{Documents}
Query Table:{Query Table}
Enriched Table:{Enriched Table}
```

In this format, {Documents} represents the relevant documents, {Query Table} and {Enriched Table} denote the serialized query table and enriched table respectively. Then, we define the template for a single query as

```
{System Prompt}
{Examples}
Documents:{Documents}
Query Table:{Query Table}
Enriched Table:
```

where the {System Prompt} is a brief instruction that contextualizes the *SDE* task for the model. During inference, we randomly pick examples selected from the training set to fill in the `{Examples}` and combine them with the query documents and query table to form the prompt for the LLM. The model is expected to output the corresponding enriched table for the given query.

For the end-to-end model, we present a simple baseline by concatenating all the retrieved documents into a single document, which reduces the problem to the text-to-table generation task. Then, we use the state-of-the-art method called *text-to-table* [48] with fine-tuning from the BART model to optimize the cross entropy loss [25]. We also employ our proposal utilizing k-NN interpolation. We train both the *text-to-table* model and our proposed method using Adam optimizer with dropout until convergence.

5.3.2 ***Metrics.*** We follow and extend the methodology adopted in prior work on related tasks [48], and we evaluate the performance of each method based on the number of correctly enriched attribute descriptions and attribute values. The evaluation is based on the recall, precision and F1 score.

Specifically, we use $O(x, y)$ to denote the similarity metric between two text cells x, y in the table. For this similarity metric, we consider three options: exact match, chrF score, which calculates character-level n-gram similarity [38], and rescaled-Bert score, which leverages BERT embeddings for text similarity [51].

Then, for each enriched table t and the corresponding ground truth table t_{gt}, we compute the precision as

$$P = \frac{1}{|t|} \sum_{c \in t} max_{c' \in t_{gt}} O(c, c'),$$

recall as

$$R = \frac{1}{|t_{gt}|} \sum_{c' \in t_{gt}} max_{c \in t} O(c, c'),$$

and f1 score as

$$\frac{2}{1/P + 1/R}.$$

Moreover, we quantify "format error" as the percentage of the output where the enriched table is in an incorrect format.

Dataset	Method	Attribute Header F1			Attribute Values F1			Format Error
		Exact	chrF	rescaled Bert	Exact	chrF	rescaled Bert	
SDE-Wikibio	**TEI**	**74.29**	**80.88**	**90.15**	**62.23**	**68.29**	**71.13**	0.0
	Text-to-table	69.10	76.91	88.19	51.38	60.16	62.25	0.0
	GPT-3.5	55.41	67.23	84.73	45.53	53.67	56.51	9.0
	Llama-2-70B	38.13	50.84	66.85	29.75	36.76	40.43	37.7
SDE-E2E	**TEI**	**97.92**	**98.36**	**98.94**	**86.62**	**88.37**	**91.28**	0.0
	Text-to-table	95.73	96.71	97.75	72.13	76.79	83.04	0.0
	GPT-3.5	90.39	92.98	96.67	69.51	74.57	83.72	4.54
	Llama-2-70B	85.85	89.22	95.47	66.29	70.78	81.76	6.21
SDE-Resume	**TEI**	**99.82**	**99.84**	**100**	**89.15**	**92.11**	**93.63**	0.0
	Text-to-table	NA	NA	NA	NA	NA	NA	NA
	GPT-3.5	98	98.23	99.85	88.98	91.55	93.55	7.86
	Llama-2-70B	82.90	85.32	97.52	62.34	69.93	82.85	64.5
SDE-Rotowire-Team	**TEI**	**84.50**	**88.09**	**88.07**	**72.64**	**75.26**	**89.17**	0.0
	Text-to-table	NA	NA	NA	NA	NA	NA	NA
	GPT-3.5	52.53	61.87	63.54	23.77	25.24	40.53	34.21
	Llama-2-70B	47.13	55.32	55.66	5.25	7.73	54.41	5.53
SDE-Rotowire-Player	**TEI**	**85.81**	**89.48**	**93.34**	**72.72**	**74.71**	**85.07**	0.0
	Text-to-table	NA	NA	NA	NA	NA	NA	NA
	GPT-3.5	55.45	65.20	75.43	19.78	22.01	34.93	30.80
	Llama-2-70B	54.38	63.63	74.41	5.30	9.11	25.47	59.25

Table 4. Table comparing the performance of our method **TEI** with existing methods across multiple datasets with retrieval dataset, based on F1 scores for attribute headers and attribute values. NA indicates the model can not generate a result for the dataset, and bold numbers indicate the best performance.

5.4 Results on the Ground Truth Dataset

First, we present results based on the ground truth version of our datasets where each instance in the dataset includes only relevant documents according to the ground truth. Therefore, any retrieval noise is eliminated from this part of the study. In Table 1, we report the F1 scores for both the attribute header and attribute value with various similar metrics. Note that each instance in the Rotowire dataset consists of two tables: Players and Teams, where we separately report the performance metrics for each table, denoted as *SDE*-Rotowire-Player and *SDE*-Rotowire-Teams.

As illustrated in Table 1, we observe no single method demonstrates superior performance compared to others. Among methods not utilizing an LLM, *text-to-table* yields higher F1 in the *SDE*-E2E dataset and in the player table of the *SDE*-Rotowire dataset, while **TEI** yields a higher F1 score in the *SDE*-Wikibio dataset. Both methods perform similarly for the team table in the *SDE*-Rotowire dataset.

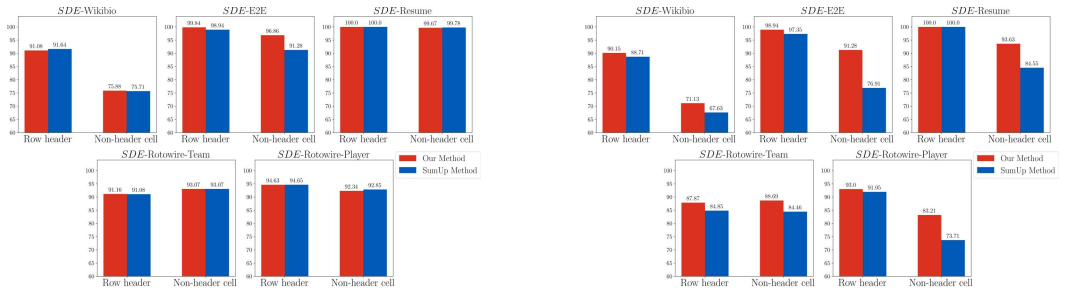
For large language models, their performance varies significantly. In most datasets, they achieve relatively low F1 scores. This is potentially because large language models often fail to identify the potential schema relevant to the task at hand, as evidenced by their low F1 scores in attribute header identification. For $SDE - Resume$, as the dataset is synthetically generated, it only contains

a limited number of schema. In this case, we can observe LLMs obtain a much higher F1 in both the attribute header and values as it can correctly extract the fixed schema based on the examples.Furthermore, we observe that in datasets with simple table formats like Wikibio and E2E, GPT-3.5 usually understands the required output format correctly. In contrast, LLAMA2-70B often produces enriched tables that do not adhere to the expected format. Nevertheless, neither model correctly understands the format requirements for the Rotowire dataset, which has more complex formatting criteria.

In short, our method consistently performs robustly, showing comparable results to the top models across various datasets and metrics. This is expected, as the documents in the datasets are noise-free. Our K-NN interpolation will assign similar weights for each of the documents. Consequently, the decoder will simply receive the average encoding of all the documents as its input. Compared to the end-to-end approach, which concatenates all documents as input, **TEI** offers an advantage as it operates on a shorter sequence length.

However, it is worth to highlight that **TEI** exhibits good performance compared to *text-to-table* while reducing the total token length by at least half. In the synthetic resume dataset, this reduction in token length is especially advantageous, where the *text-to-table* cannot enrich the table due to token constraints by the pre-trained language model. In contrast, our method overcomes this limitation.

Moreover, the reduced rate in sequence length presents an interesting trade-off between performance and both memory and computational requirements. In our experiments, we found that increasing the number of relevant documents (thereby increasing the reduced rate) negatively impacts the final F1 score. Therefore, to optimize the final F1 score, it is often beneficial to merge smaller documents, enabling more effective utilization of the constrained sequence length.



(a) Ground-Truth Datasets (b) Retrieval Datasets

Fig. 5. Comparison between our method **TEI** with the SumUp method on the Ground-Truth datasets and Retrieval datasets. The F1 Score based on the rescaled-Bert Score is reported in the graph

5.5 Results on Retrieval Dataset

Now, we will consider the result in the case where the relevant documents for a given query table are provided by the retrieval model. As shown in Table 4, in this setting, our method **TEI** outperforms all other alternatives across every dataset by a margin of 10% in terms of F1 score. Compared to the results on the ground truth dataset, this success demonstrates that **TEI** can more effectively identify and denoise irrelevant documents identified by the retrieval model than other methods.

It should be noted that for the WikiBio and E2E datasets, the *text-to-table* model achieves comparable results in attribute headers but significantly lower F1 scores for attribute values. This

indicates that although the end-to-end model can effectively identify the potential schema related to the text and query, it fails to retrieve the correct values associated with each attribute in the presence of noisy information. Furthermore, as the number of candidate documents for each query table increases, the model can not augment the Rotowire dataset due to sequence length limitations. In contrast, **TEI** maintains nearly constant sequence length regardless of the number of candidate documents, making our method more scalable.

For methods utilizing an LLM, their F1 score does not drop significantly compared to the *text-to-table* model. This finding demonstrates that LLMs can effectively identify documents relevant to the query table, even when those documents contain a mixture of pertinent and irrelevant information. The comparable performance between GPT-3.5-turbo and **TEI** in the synthetically generated *SDE*-Resume dataset further proves that once the LLM is able to identify a suitable schema, it has the potential ability to extract the correct values from noisy input documents. However, for all other real-world datasets, LLM are still unable to extract a relevant schema from the document. Thus the performance remains lower than that of the *text-to-table* model and that of our method.

5.6 Ablation Study

We wish to evaluate the impact of two major design choices for **TEI**, namely the effectiveness of our proposed interpolation mechanism compared to other options and the specific choice of the queries in the problem formulation.

5.6.1 ***Effect of K-NN Interpolation.*** We conduct an ablation study on our method **TEI** by replacing the *Knn-Interpolation* component with the summation of encodings from all input documents. We refer to this alternative design as *SumUp* method. Figure 5 illustrates the F1 score based on rescaled-Bert for ground-truth datasets and retrieval datasets.

We expect that **TEI** and the *SumUp* method perform similarly in the ground truth dataset as our interpolation will assign similar weights for each of the documents. As is evident in figure 5a, our method and the *SumUp* method perform comparably with less than 1% difference in F1 score in almost all the datasets except the E2E dataset.

In the E2E dataset, **TEI** is on par with *SumUp* for the case of inferring attributes (*Row Header* in Figure 5a) and performs slightly better than *SumUp*, with a 5% increase in attribute value (*Non-header cell*) F1 score. The superior performance is mainly attributed to the imbalance in term document length in the E2E dataset, where longer documents often contain richer information than shorter ones and hence are more important for augmentation. To illustrate, consider two example documents from the E2E dataset: *"Alimentum is a French venue located on the riverside. Located in the riverside area the Alimentum is kid friendly."* and *"Alimentum is located in riverside"*, where the first document contains much more information than the second. The result on E2E dataset demonstrates that **TEI** is capable of correctly prioritizing more informative documents for augmentation.

For retrieval datasets, as evident in Figure 5b we observe a *significant accuracy improvement* between **TEI** and the *SumUp* method. In particular, **TEI** is generally better than alternatives when inferring attribute values (*Row Header* in Figure 5b) and offers accuracy advantages of more than 15% when inferring attribute values (*Non-header cell* in Figure 5b).

This underscores the necessity of using *Knn-Interpolation* when dealing with noisy documents.

5.6.2 ***Effect of Table Normalization.*** In our ongoing investigation thus far, we have established the query formulation for the *SDE* problem as a set of tabular entities. In essence, we have posited our desire to curate an enriched, potentially enhanced, collection of *normalized tables* as the desired output.

Method	F1-Score (%)	Relative F1 Score(%)
Text-to-table		
Attribute Header	85.41	-3.51
Attribute Value	83.68	-2.34
TEI		
Attribute Header	87.98	-0.94
Attribute Value	85.10	-0.30

Table 5. The exact F1 score for denormalized *SDE*-Rotowire table with ground truth documents. The relative F1 score shows the different from the normalized *SDE*-rotowire.

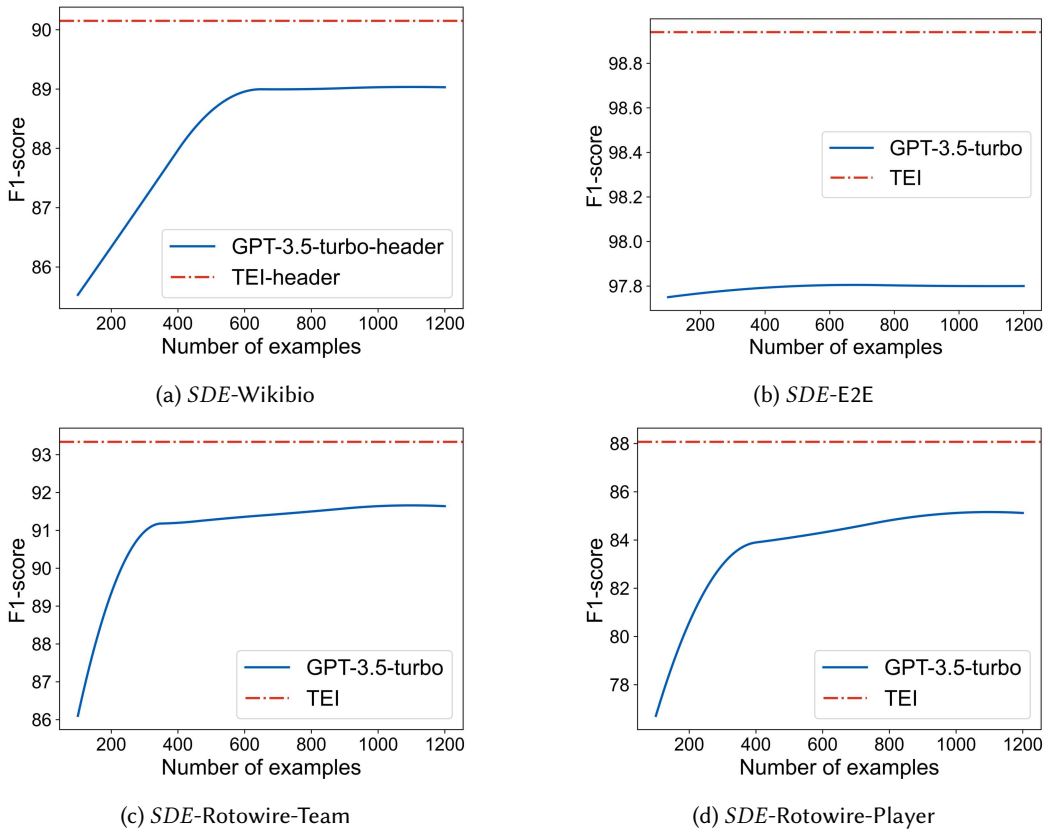
Method	F1-Score (%)	Relative F1 Score(%)
TEI		
Attribute Header	84.81	-0.34
Attribute Value	76.12	+3.44

Table 6. The exact F1 score for denormalized *SDE*-Rotowire table with retrieval documents. The results for the *text-to-table* method have been omitted due to the constraints on content length imposed by its model.

We examine the merits of this design choice in the table enrichment processes. To commence, we embark on the creation of a denormalized *SDE*-Rotowire dataset, achieved by amalgamating the Team table with the Player table. Concretely, for each entry in the Player table, we establish an association with a corresponding entry in the Team table, and thus, we forge the join version of these two tables. This yields a substantial table replete with expected redundancy in attributes pertaining to teams. Given that the attributes in the *SDE*-Rotowire datasets are predominantly numeric, such as scores, our evaluation centers around the F1 score measuring exact matches for this denormalized Rotowire dataset against ground truth documents.

Table 5 presents the outcomes for **TEI**, and the *text-to-table* approach, alongside their relative performance compared to their normalized counterparts for datasets augmented with ground truth documents. Our observations reveal that both models exhibit a slightly diminished performance in the denormalized context when compared to their normalized counterparts. This phenomenon is anticipated since denormalized tables introduce a greater number of attributes, necessitating the models to account for inter-relationships among the attributes. However, it is noteworthy that in this scenario, **TEI** outperforms the *text-to-table* approach, with less than a 1% reduction in exact F1, compared to a 2-3% drop for *text-to-table*.

The superior performance demonstrated by **TEI** in the denormalized context can be attributed to its query table encoding during interpolation. Specifically, the encoding of queries derived from a single, albeit larger, table exhibits greater semantic coherence and, consequently, is more representative than queries derived from multiple smaller tables. This finding gains further support from the results depicted in table 6, which illustrate that **TEI** achieves an even higher F1 Score, with a 3% increase in attribute values, when dealing with documents containing noisy information. These results substantiate the assertion that **TEI** excels in handling denormalized tables and showcases resilience across diverse table structures.



(a) *SDE*-Wikibio

(b) *SDE*-E2E

(c) *SDE*-Rotowire-Team

(d) *SDE*-Rotowire-Player

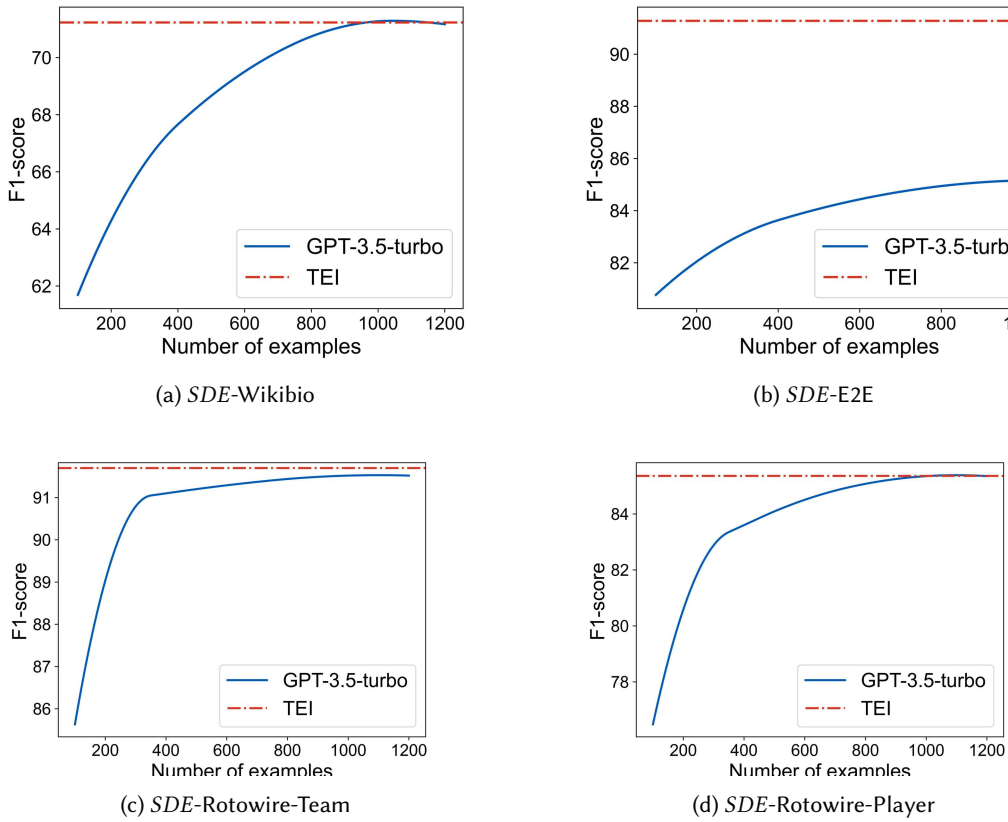
Fig. 6. Comparison between fine-tuned GPT-3.5-turbo on attribute headers with F1 for the rescaled-Bert score.

5.7 Finetuning the LLM

5.7.1 ***Fine-tuning LLMs for SDE.*** In this section, we explore fine-tuning of LLM utilizing datasets in our study. By fine-tuning an LLM we should be able to enhance the capabilities of LLM to provide accurate answers to the problem of interest in this study. It has been demonstrated [31] that compared to few-shot learning, fine-tuning an LLM offers more precise control over a model's actions and increased accuracy and robustness by training on many more examples, achieving better results on tasks like natural language understanding, sentiment analysis, etc. However, fine-tuning also incurs high computation costs as well as substantial memory requirements.

We first illustrate the results of fine-tuning the best-performing LLM, GPT-3.5-turbo by utilizing three real-world SDE datasets, namely *SDE*-Wikibio, *SDE*-Rotowire, and *SDE*-E2E; the relevant documents for a given query table are provided by the retrieval model. To better understand model behaviour during fine-tuning, we fine-tune by systematically increasing the number of training samples until it converges, monitoring the model performance based on the F1 of the rescaled-Bert Score. In Figure 7 and 6, we compare the result of the augmented header and corresponding values of the fine-tuned GPT-3.5-turbo with the results obtained using **TEI**.

[1]For the *SDE*-Rotowire dataset, since the LLM tends to generate hallucinated data, we have substituted all the empty cell values with "None". This ensures that both models are equally penalized for creating unfounded or empty values.



(a) *SDE*-Wikibio (b) *SDE*-E2E

(c) *SDE*-Rotowire-Team (d) *SDE*-Rotowire-Player

Fig. 7. Comparison between fine-tuned GPT-3.5-turbo on attribute values[1] with F1 for the rescaled-Bert score.

As shown in the figure, both models achieve similar F1 score on attribute values on both *SDE*-Wikibio and *SDE*-Rotowire datasets. For attribute header however, **TEI** consistently achieves better F1 than fine-tuned GPT-3.5-turbo. By closer inspection of the results we observe, that tables augmented by GPT-3.5-turbo often include *incorrect* but seemingly relevant headers (which are results of hallucinations) which appear plausible but not actually present in the source data; in some cases, these headers are added with all corresponding values being empty. Thus, even though GPT-3.5-turbo is exhibiting good performance, we hypothesize that there might be data leakage for these two datasets as the information for those datasets is available publicly. This is further supported by the fact that the output of the GPT-3.5-turbo occasionally contains information that correctly pertains to the query table while the information does not exist in the relevant input documents. For instance, when augmenting a table about King Ekathotsarot of Ayutthaya, the model not only included information from the input documents but also included his birth date and royal house, which are details not present in the retrieved documents. Nevertheless, fine-tuning GPT-3.5-turbo is improving the model's ability to identify a suitable data schema and extract relevant information.

For *SDE*-E2E dataset, however, the fine-tuned GPT-3.5-turbo performs worse than **TEI** with an 8% lower F1 score. After a manual comparison of the models' outputs, we attribute the performance gap primarily to GPT-3.5-turbo inability to accurately identify relevant documents for the given query table in this particular dataset. For instance, when seeking information about the *A Slice Of Heaven*

Model	Fine-tuning Cost	Inference Cost
GPT-3.5-turbo	240	3300
TEI	1.93	119.44

Table 7. Estimated fine-tuning cost in USD with a dataset of 10,000 samples and inference on the model with 1 million queries. The cost for GPT-3.5-turbo is calculated based on its API cost [2]. For TEI, we estimate the cost by calculating the total GPU hours utilized and then applying the GPU rental price [3]to determine the final cost.

coffee shop located in Riverside, GPT-3.5-turbo also erroneously includes information about other coffee shops, which are included in the retrieved documents but constitute false positives (irrelevant information). We postulate that this is due to the unique challenges present in the E2E dataset, which includes conflicting information. For instance, two different reviews from different sources for the same restaurant. Specifically, the training dataset from *SDE*-E2E only contains relevant reference documents with conflicting information. However, at query time irrelevant information (false positives) and conflicting information (as observed at query time) coexist. For example, the retrieved documents contain a false positive restaurant result along with rating information. This likely confounds GPT-3.5-turbo, affecting its accuracy in document identification since it cannot differentiate between correct and false positive results to obtain the ratings. However, in **TEI**, we first use *Knn-Interpolation* to pre-filter the irrelevant information (false positives) at the encoder step. As *Knn-Interpolation* is trained in a contrastive manner, it operates effectively without the need of explicit supervised training to understand and filter irrelevant documents. Consequently, the decoder will receive embeddings with minimal impact from irrelevant information and only need to deal with conflicting information when generating the enriched table, a scenario for which the model has been specifically trained.

5.7.2 ***Fine-tuning Cost.*** In the preceding section, we have demonstrated that fine-tuning a Large Language Model (LLM) improves its ability to recognize appropriate data structures and gather relevant information for SDE tasks. However, this comes at a higher cost compared to **TEI**. Notice that in order to utilize a fine-tuned LLM one has to incur costs to fine-tune the LLM (a relatively minor cost) but also incur ongoing inference costs. Sample expenses associated with fine-tuning and inference for both methods are summarized in Tables 7 and 8.This comparison highlights that the costs incurred by GPT-3.5-turbo are significantly higher than those for **TEI**. In particular, it is evident that utilizing a fine-tuned state-of-the-art LLM costs approximately two orders of magnitude than **TEI** both for fine-tuning and inference. Additionally, the inference for GPT-3.5-turbo has to be conducted through an API call which is significantly slower than the direct methods employed by **TEI**. This further emphasizes the efficiency and scalability of **TEI** in terms of both cost and performance.

5.7.3 ***Fine-tuning a Smaller LLM.*** A natural question to ask is whether fine-tuning a smaller-scale Large Language Model can offer a better balance between cost and performance. To explore this, we experimented with fine-tuning llama2-7B using Low-Rank Adaptations (LoRA [18]). This approach allowed the fine-tuning process to be conducted on a single Nvidia A100 GPU with 40GB of GPU memory. By utilizing a less resource-intensive model and an efficient fine-tuning technique, we aimed to assess whether this setup could yield competitive performance while

[2]https://openai.com/pricing#language-models
[3]1.29 USD/hour for NVIDIA A100 on Lambda https://lambdalabs.com/

TEI	text-to-table	LLama2-7B	GPT-3.5-turbo	LLama2-70B
3	3.2	0.1	0.5	0.3

Table 8. Inference time for different models expressed as query per second (higher is better). For GPT-3.5-turbo and Llama2-70B the inference is conducted through an API call, where the run time for **TEI**, *text-to-table* and Llama2-7B are measured using a single local Nvidia A100 GPU.

Model	Attribute Header	Attribute Value	Format Error
llama2-7B	80.99	59.30	34%
TEI	90.15	71.13	0%

Table 9. Fine-tuning result on smaller-scale LLM, Llama2-7B, evaluated based on the F1 for the rescaled-Bert score.

reducing fine-tuning costs, memory requirements as well as inference costs to be comparable to **TEI**. We utilize the *SDE*-Wikibio dataset with the retrieved documents to verify the efficiency of the fine-tuned model. The results shown in Table 9, however, demonstrate that such an approach does not achieve an F1 score comparable to that of **TEI**. The inferior accuracy can be attributed to two primary factors. Firstly, the smaller-scale LLM had difficulties in recognizing the specific structure of the augmented table, leading to over 30% of its output being misaligned. Secondly, and perhaps more critically, the model exhibited a tendency to produce repetitive information in its outputs. For instance, it frequently repeats attribute names and family names for the same entity. Therefore, to enhance the effectiveness of the smaller-scale LLM, critical optimization beyond basic fine-tuning is necessary. This is, in our view, an interesting direction for future exploration.

6 RELATED WORK

6.1 Data Discovery

Much recent research has been conducted on automating data discovery procedures within data lakes of structured data (tables). Those tasks include unionable table search, Joinable table search, schema matches, etc. [3, 21]. Systems that are built for those tasks leverage a wide variety of similarity signals such as schema-based match, value-based match, knowledge graphs match, and embedding-based match. More recent systems utilize a combination of these indicators, such as TURL, Aurum, D3L [3, 9, 14].

Contrary to a structured data lake, Data Discovery over Unstructured Data Lake has received less attention in the data management community, mainly due to the challenge of integrating information from two modalities of structured and unstructured data. Most techniques rely on transforming structured tables into unstructured data and utilizing language models to discover the relation between tables and documents [4]. Most recent work [12], proposed an embedding-based joint representation for multi-modal data sets and built up a holistic pipeline to support Unstructured data. However, all these works mainly focus on finding the relevant document to the given query without integration, which is orthogonal to this work.

6.2 Information Extraction

Information Extraction (IE) refers to the automated process of identifying and extracting structured information from unstructured or semi-structured data sources, such as text documents. This