Reimagining Retrieval Augmented Language Models for Answering Queries

[Reality Check Theme Track]

Wang-Chiew Tan Yuliang Li Pedro Rodriguez
Richard James* Xi Victoria Lin Alon Halevy Scott Yih
Meta
{wangchiew,yuliangli,victorialin,ayh,scottyih}@meta.com rich@richjames.co*

Abstract

We present a reality check on large language models and inspect the promise of retrieval-augmented language models in comparison. Such language models are semi-parametric, where models integrate model parameters and knowledge from external data sources to make their predictions, as opposed to the parametric nature of vanilla large language models. We give initial experimental findings that semi-parametric architectures can be enhanced with views, a query analyzer/planner, and provenance to make a significantly more powerful system for question answering in terms of accuracy and efficiency, and potentially for other NLP tasks.

1 Introduction

As language models have grown larger (Kaplan et al., 2020; Hoffmann et al., 2022), they have fared better and better on question answering tasks (Hendrycks et al., 2021) and have become the foundation of impressive demos like ChatGPT (Ouyang et al., 2022; ChatGPT3-OpenAI). Models like GPT-3 (Brown et al., 2020) and ChatGPT generate fluent, human-like text, which comes the potential for misuse as in high-stakes healthcare settings (Dinan et al., 2021). Large language models (LLMs) also come with several significant issues (Hoffmann et al., 2022; Bender et al., 2021).

LLMs are costly to train, deploy, and maintain, both financially and in terms of environmental impact (Bender et al., 2021). These models are also almost always the exclusive game of industrial companies with large budgets. Perhaps most importantly, the ability of LLMs to make predictions is not commensurate with their ability to obtain insights about their predictions. Such models can be prompted to generate false statements (Wallace et al., 2019a), often do so unprompted (Asai et al., 2022) and when combined with its ability to easily fool humans, can lead to misuse (Macaulay, 2020).

In recent years, we have seen the promise of retrieval-augmented language models partially addressing the aforementioned shortcomings (Guu et al., 2020; Lewis et al., 2020; Borgeaud et al., 2021; Izacard et al., 2022; Yasunaga et al., 2022a). The architecture of such models is *semi-parametric*, where the model integrates model parameters and knowledge from external data sources to make its predictions. The first step of performing a task in these architectures is to retrieve relevant knowledge from the external sources, and then perform finer-grained reasoning. Some of the benefits these architectures offer are that the external sources can be verified and updated easily, thereby reducing hallucinations (Shuster et al., 2021a) and making it easy to incorporate new knowledge and correct existing knowledge without needing to retrain the entire model (Lewis et al., 2020). Models that follow semi-parametric architectures (SPA) are typically smaller than LLMs and they have been shown to outperform LLMs on several NLP tasks such as open domain question answering (see Table 1). Recent work that extends LLMs with modular reasoning and knowledge retrieval (Karpas et al., 2022; LangChain) is also a type of SPA.

In this paper we argue that building on the core ideas of SPA, we can potentially construct much more powerful question answering systems that also provide access to multi-modal data such as image, video and tabular data. We describe POSTTEXT, a class of systems that extend SPA in three important ways. First, POSTTEXT allows the external data to include *views*, a concept we borrow from database systems (Garcia-Molina et al., 2008). A *view* is a function over a number of data sources, $V = f(D_1, ..., D_n)$. In databases, SQL queries are used to define tabular views. For example, V can be a table of records of minors that is derived from a table of person records by selecting only those with age<18. In general, however, views need not be tabular. When a view is materialized

Model	#Params	Outperformed LLM's sizes	Tasks
REALM (Guu et al., 2020)	330M	11B (T5)	Open-QA
RETRO (Borgeaud et al., 2021)	7.5B	178B (Jurassic-1), 280B (Gopher)	Language modeling
Atlas (Izacard et al., 2022)	11B	175B (GPT-3), 540B (PaLM)	Multi-task NLU, Open-QA
RAG (Lewis et al., 2020)	400M	11B (T5)	Open-QA
FiD (Izacard and Grave, 2021)	770M	11B (T5), 175B (GPT-3)	Open-QA

Table 1: The sizes of SPA models with those of comparable or outperformed LLMs.

(i.e., executed and stored), it may be useful for answering certain queries[1] more effectively. In this paper, we adopt a more general notion of views, not limited to results of SQL queries, which can (compositionally) support a variety of user questions. Views are particularly important to support multi-modal data, because combinations of data from multiple modalities can be modeled as views. Second, POSTTEXT contains a question analyzer and planner module that decides on the best strategy to answer a question that may involve first answering multiple subquestions in sequence or in parallel. This module bears similarity to query optimization techniques in database systems but will go significantly beyond the techniques established in database systems since, there are multiple different ways to answer a natural language question, especially with the availability of multi-modal data. Finally, POSTTEXT supports computing the provenance of answers to questions. The provenance-aware answer generator module can track the evidence (training data or external sources) that is used for the answers, even if views are used as intermediate results.

We illustrate the power of POSTTEXT with examples in the next section and also the overview of its architecture. In the remaining sections, we describe the different components of POSTTEXT.

2 Overview of PostText

Example 1 Consider a setting where we answer questions over data that includes images of dishes and text with restaurant reviews. We can create a view that aligns these two data sets so we can answer more complex queries readily. The view, the table in the middle of Figure 1, aligns dishes with relevant reviews and the corresponding restaurants. Note that creating this view involves an intermediate step of identifying the name of the dish in an image. The view also stores the provenance links to the actual reviews from which the snippets were extracted. There are also provenance links for the images and the name of the dish (not shown).

This view can be used to answer questions that would be more difficult without it. For example, if a person recalls a nice dish she had in the past but does not remember its name and is trying to figure out which restaurants serve the same dish and what are the reviews, she can pose the question, which includes both the question in text and an image of the dish. The answer states the name of the dish in question and lists restaurants with top reviews for that dish, along with images of the dish and snippets of those reviews and their provenance.

Example 2 The same view can also be used to answer the question "*how many reviews raved about Shaking beef?*". The answer requires counting the number of reviews that are synonymous to very positive reviews about Shaking beef. The view surfaces the reviews associated with Shaking beef immediately and alleviates the amount of work that is required to compute the answer otherwise.

The examples show that some questions can be answered more easily if they are supported by views that surface useful associations between data. In fact, indices are a type of views to accelerate lookups between an item and its attributes. In database systems, views have been used extensively to enable more efficient query answering (Halevy, 2001; Goldstein and Larson, 2001) with significant work on automatically materializing a set of indices for efficient query answering (Jindal et al., 2018; Das et al., 2019). A set of views and indices are defined automatically or manually in anticipation of a set of frequently asked queries under a budget constraint, e.g., space, so that during runtime, most of the incoming queries can be answered immediately or after applying simple operations over the views. Otherwise, the system falls back to answering the queries using the actual data sources. In other words, POSTTEXT prefers to use views to answer the questions, which will likely to be more efficient and accurate in general but otherwise, the system falls back to the traditional question answer-

[1]We use queries and questions interchangeably.

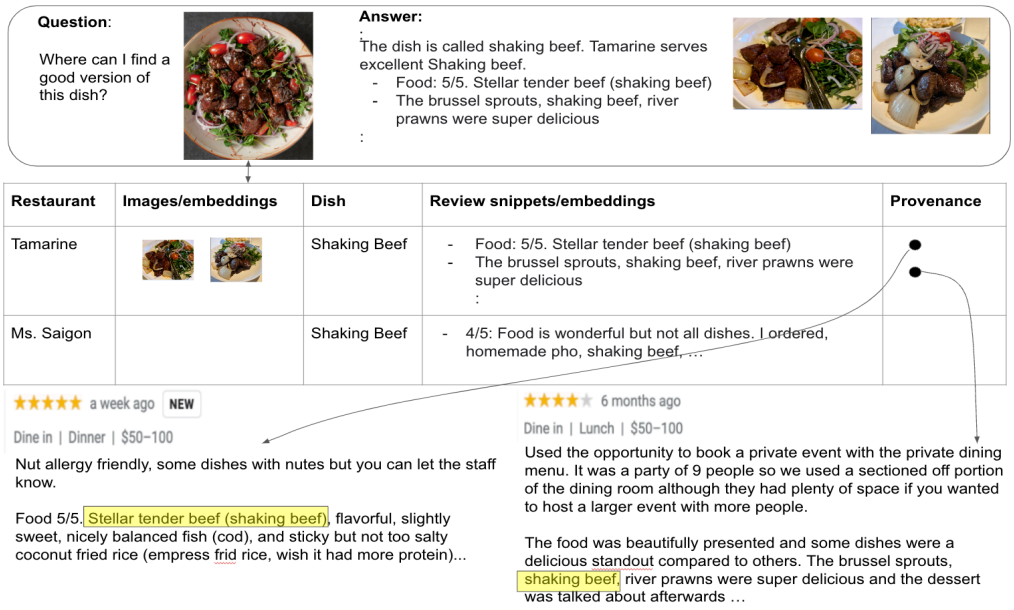

Figure 1: Multimodal question with multimodal answer. The view (middle) associates the dishes with its corresponding review snippets and images. The provenance links show where the snippets are extracted from. There are also provenance links for the images and name of the dish (not shown).

ing strategy. In addition to query answering, views have also been used to define content-based access control (Bertino and Sandhu, 2005), i.e., which parts of the data are accessible and by whom.

The examples also show how provenance is provided as part of the answer. In these examples, it happened that provenance was easily determined through the provenance links that are already captured in the views. If actual data sources are accessed, the links to the data sources used (e.g., spans of text documents, parts of images, segments of videos) to derive the answer are provided as part of the answer. If the answer is generated by the language model, we trace how POSTTEXT derives the answer from parametric knowledge and retrieved data through analyzing its weights or determining "influential" parametric knowledge (Section 6) similarly to (Akyürek et al., 2022).

PostText architecture POSTTEXT enhances the core architecture of semi-parametric models with three components: views, a query analyzer & planner (QAP), and a provenance-aware answer generator (PAG). In addition, all components including the "traditional" knowledge retrievers are equipped to manage both structured and unstructured data of different modalities.

Figure 2 shows the architecture of POSTTEXT. Views are synthesized from different types of external data sources (e.g., text, images, videos, and tabular data), which can be public or private. When a question is posed in natural language (NL), the QAP module interprets and decomposes the question into subquestions whose answers can be composed to obtain an answer to the input question. QAP coordinates with the knowledge retriever to derive the data needed to answer these questions. It also coordinates with the PAG module with its plan so that provenance-aware answers can be returned.

Adding these components raises interesting challenges such as what views should we construct and how do we construct and maintain these views automatically as data sources changes? What is a good plan for deriving an answer and how do we choose among alternative plans? And how do we measure the "goodness" of an answer with provenance?

In the remaining sections, we describe the challenges associated with each of these components

3 Data Sources and Views

Data Sources Most existing work on retrieval augmented language models are focused on text. More recently, (Chen et al., 2022; Yasunaga et al., 2022b; Sheynin et al., 2022) has applied SPA models on image-text and text-only corpus. The data sources in POSTTEXT are multi-modal, unstructured or structured. They can be external public data sources or private ones.

Views Views are results computed (not necessarily through SQL queries) from data sources or other

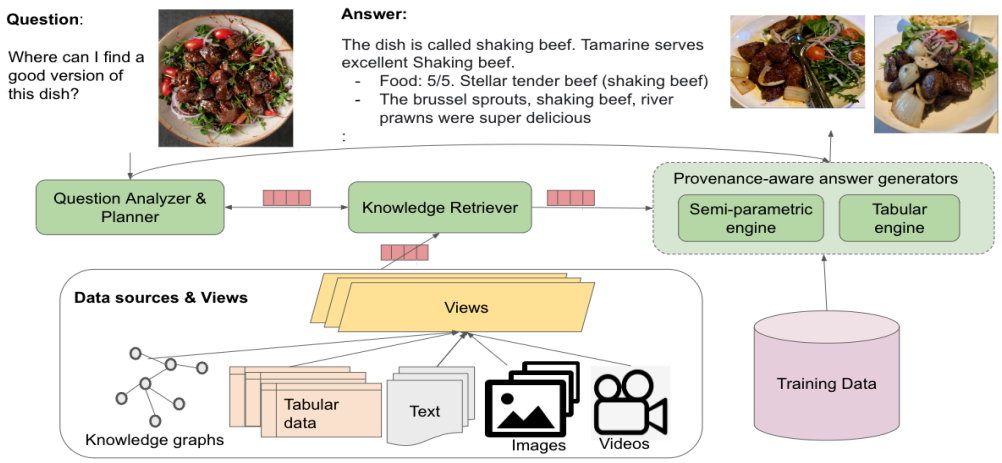

Figure 2: Semi-parametric architectures enhanced with views, a query analyzer & planner module, and a provenance-aware answer generator. The data sources may be public or private.

views. For example, a view can be a document involving data of different modalities (e.g., an image or a table). Views are powerful constructs for surfacing important and useful associations that are not obvious otherwise, whether they are associations from data within one data source or across multiple data sources. The table in Figure 1 is a view over restaurant reviews from Yelp, Google, and images provided by restaurants. This view makes it easier to compute the number of reviews associated with each dish in each restaurant or even across all restaurants. This view also makes it easier to determine the answer as to which dishes has more reviews than Shaking beef at Tamarine.

Indexes are a special type of views. They associate an item with its attribute. Several implementations of retrieval augmented language models (Guu et al., 2020; Lewis et al., 2020; Izacard et al., 2022) already construct indices that associate a document with its nearest neighbors. Recently, GPT-index (GPT-Index, 2022) developed a set of APIs for creating data structures that can be traversed using LLMs to answer queries. The data structures are structured indexes and can be used to determine an answer to a question.

Relational views are extensively used in data warehouses for optimizing queries. Indexes and views are typically created by users or database administrators or they can be automatically selected (Agrawal et al., 2000; Schnaitter et al., 2007; Jindal et al., 2018) and tuned (Agrawal et al., 2006; Bruno and Chaudhuri, 2008) to efficiently answer queries of a given workload (Das et al., 2019), which are queries that are anticipated to be frequently occurring. In typical settings, a set of views are constructed, usually under a budget constraint such as space, to maximize the queries that can be answered (either directly or through applying a few simple operators on the views) in a given workload. When a new query arrives after the views are constructed, the query optimizer determines the best plan to adopt for computing the answer. Queries are directly executed over the views if possible. Otherwise, it falls back to old strategy of answering the query with the data sources. For example, early last year, in anticipation of frequent queries about statistics of past World Cups due to the World Cup 2022 event at the end of the year, a set of views about the different World Cup statistics could have been constructed a priori so that most World Cup related questions can be directly answered using the views.

We hypothesize that views in POSTTEXT can bring similar benefits to question answering. The right views will make it easier for the QAP module and the knowledge retriever to discover and obtain relevant data and subsequently for the answer generator to derive the right answers. Existing SPAs (Guu et al., 2020; Lewis et al., 2020; Izacard et al., 2022) are already leveraging dense-vector indices to accelerate the retrieval of document spans. In POSTTEXT with views being available, it is a natural extension to annotate each view with a description of its content (e.g., "*Restaurants and highly ranked dishes*"), which would make it even easier for the knowledge retriever to find the relevant data. The core challenges in developing views are how do we determine what is a "right" set of views to materialize automatically or semi-automatically? How do we incrementally maintain such views as

data sources are updated? These problems are extensively studied in the database community and it will be interesting to explore those ideas that transfer to the POSTTEXT.

The architecture can also be instrumented in such a way that views are the only sources of data for the knowledge retriever (i.e., actual data sources are excluded). Hence, in this case, views act as a gateway that define which parts of the data sources are accessible by the knowledge retriever to answer queries. Finer-grained access control can also be instrumented through views as described in (Bertino and Sandhu, 2005). With views, it is also possible to enable a finer-grained public-private autoregressive information retrieval privacy system (Arora et al., 2022).

4 Question Analyzer & Planner

The question analyzer and planner (QAP) module examines the input question and generates a plan, i.e., a sequence of sub-questions whose answers can be combined to form an answer to the input question. For each subquestion in the plan, QAP first checks whether external knowledge is needed. If not, the language model can be used to derive the answer. Otherwise, the subquestion is passed to the knowledge retriever to discover and retrieve relevant data for the subquestion at hand. The results from the knowledge retriever and the plan are passed to PAG (i.e., the rightmost green box in Figure 2). It is still an open and challenging question to determine whether a language model can confidently answer a question (Kamath et al., 2020; Si et al., 2022). Any solution to this problem will help improve the plan generator.

An example plan from the QAP module for our running example is as follows: (1) find the name of the dish X in the input image, (2) find restaurants that serve X, (3) find the top restaurant among the results from (2). This plan is viable because (a) there is an index associating embeddings of images with the name of the main entity of the image, (b) there exists a view as shown in Figure 1, which supports the search for restaurants that serve a particular dish. Top answers can be derived by computing the scores of the reviews or approximating it based on the sentiment of the reviews and then ranking the results based on such scores. The information from (2) is passed to PAG which will compute the answer along with its provenance. This plan is based on the heuristic to push selection conditions early before joining/combining different data sources if needed. The conditions in the question are "good version" and "this dish". In this case, no joins are required as the view already combines the required information in one place. Hence, QAP seeks to first find the name of the dish to narrow down the reviews restricted to this dish. Alternatively, it could also retrieve all good reviews before conditioning on the name of the dish. Yet another plan could be to match the image directly to the images of the view to find the top reviews. Or, it may decide to directly retrieve only top reviews with images similar to the image in the question from the external data sources and condition the answer based on the name of the restaurant mentioned in the reviews.

In all possible plans, the knowledge retriever is responsible for discovering and retrieving the relevant data for the QAP plan. In addition to the logic that may be needed for decomposing the question into subquestions, a plan is also needed for composing the subanswers obtained to form an answer to the input question. The plan is shared with the PAG module for deriving the associated provenance.

A fundamental challenge in developing the QAP module is how to derive candidate plans and decide what is the "best" plan for answering the question when there are different ways to obtain an answer. Achieving this requires understanding how to compare amongst alternative plans for deriving an answer to the question. This problem bears similarity to query evaluation techniques for database systems (e.g., (Graefe, 1993)). It will be interesting to investigate whether database query planning techniques and ideas can synergize with question understanding and planning techniques (e.g., (Wolfson et al., 2020; Dunietz et al., 2020; Zhao et al., 2021; Xiong et al., 2021) to develop a comprehensive query planner. Emerging work such as chain of thought reasoning (Wei et al., 2022), where a sequence of prompts are engineered to elicit better answers, ReAct (Yao et al., 2022), where reasoning and action techniques are applied for deriving an answer, and more recently, work that generates a plan which can call LMs for resolving subquestions (Cheng et al., 2022) are also relevant. These techniques so far are restricted to text and does not compare among different plans.

Another challenge in the context of NL questions is that while there is a single correct answer to an SQL query over a database, there are po-

tentially many different correct answers to a NL question (Si et al., 2021; Min et al., 2020; Chen et al., 2020). Hence the space of possible plans to derive the "best" answer most efficiently is even more challenging in this case.

We are advocating for a system that can reason and compare at least some viable strategies to arrive at a best plan for deriving a good answer efficiently. Naturally, one can also train a LM to create a plan. Our belief is that taking a more systematic route to planning can relief the need for the amount of training data required and will also aid provenance generation through its ability to describe the steps it took and the sources of data used in each step to generate an answer. As we shall explain in Section 5, the cost and accuracy of knowledge retrievers can also play a role in determining what is a better strategy for computing a good answer.

5 Knowledge Retriever

The role of the knowledge retriever is to provide the information that the system lacks in order to fulfill the given task, typically at the inference time. More importantly, we envision that the knowledge retriever proposed in our framework has the ability to access knowledge stored in different sources and modalities, retrieve and integrate the relevant pieces of information, and present the output in a tabular data view. The structured output contains raw data items (e.g., text documents, images or videos) and and optionally different metadata, such as textual description of each data item. Such structured output allows downstream (neural) models to consume the retrieved knowledge efficiently and also allows developers and users to validate the provenance conveniently. Existing information retrieval models mostly focus on a single form of data. Below we first describe briefly how knowledge retrieval is done for unstructured and structured data. We then discuss the technical challenges for building a unified knowledge retriever, as well as recent research efforts towards this direction.

Retrievers for unstructured data For unstructured data, such as a large collection of documents (i.e., text corpus) or images, knowledge retrieval is often reduced to a simple similarity search problem, where both queries and data in the knowledge source are represented as vectors in the same vector space (Turney and Pantel, 2010). Data points that are *close* to the query are considered as *relevant* and thus returned as the knowledge requested. Traditional information retrieval methods, whether relying on sparse vector representations, such as TFIDF (Salton et al., 1975) and BM25 (Robertson et al., 2009), or dense representations, such as LSA (Deerwester et al., 1990), DSSM (Huang et al., 2013), DPR (Karpukhin et al., 2020), are the canonical examples of this paradigm. Notice that the vector space model is not restricted to text but is also applicable to problems in other modalities, such as image tagging (Weston et al., 2011) and image retrieval (Gordo et al., 2016).

Retrievers for structured data When the knowledge source is semi-structured (e.g., tables) or structured (e.g., databases), the query can be structured and allows the information need to be defined in a more precise way. Because the data is typically stored in a highly optimized management system and sometimes only accessible through a set of predefined API calls, the key technical challenge in the knowledge retriever is to formulate the information need into a formal, structured query. To map natural language questions to structured queries, semantic parsing is the key technical component for building a knowledge retriever for structured data. Some early works propose mapping the natural language questions to a generic meaning representation, which is later translated to the formal language used by the target knowledge base through ontology matching (Kwiatkowski et al., 2013; Berant et al., 2013). Others advocate that the meaning representation should be closely tight to the target formal language (Yih et al., 2015), such as SPARQL for triple stores. Because of the success of deep learning, especially the large pre-trained language models, semantic parsing has mostly been reduced to a sequence generation problem (e.g., Text-to-SQL). For example, RASAT (Qi et al., 2022) and PICARD (Scholak et al., 2021), which are generation models based on T5 (Raffel et al., 2020), give state-of-the-art results on benchmarks like Spider (Yu et al., 2018) and CoSQL (Yu et al., 2019).

Towards a unified knowledge retriever As knowledge can exist in different forms, a unified knowledge retriever that can handle both structured and unstructured data in different modalities is more desirable. One possible solution for realizing a unified retriever is to leverage multiple single-source knowledge retrievers. When a query comes in, the QAP module first decomposes it into

several smaller sub-queries, where each sub-query can be answered using one component knowledge retriever. The results from multiple knowledge retrievers can be integrated and then returned as the final output. However, several technical difficulties, including how to accurately decompose the question and how to join the retrieved results often hinder the success of this approach. Alternatively, unifying multiple sources of information in a standard representation, using text as a denominator representation, has been promoted recently (Oguz et al., 2022; Zeng et al., 2022). If all data items have a corresponding textual description, it is possible for the knowledge retriever to use only text-based retrieval techniques to find relevant data items once all input entities of non-textual modality have been mapped to their corresponding textual descriptions.

Such approach circumvents the complexity of managing multiple knowledge stores in different format. Moreover, with the success of large multilingual and multi-modal language models (Conneau and Lample, 2019; Aghajanyan et al., 2022), data of different structures or from different modalities can naturally share the same representation space. While unifying multiple sources of information through representation learning seems to be a promising direction, it should be noted that certain structured information may be lost in the process. For example, by flatting a knowledge graph to sequences of (subject, predicate, object) triples, the graph structure is then buried in the textual form. Whether the information loss limits the retriever's ability to handle certain highly relational queries remains to be seen.

6 Provenance-aware answer generators

6.1 Semi-Parametric Engine

Demonstrating the provenance of a QA model prediction should center on identifying the data—whether in training data, retrieval corpora, or input—that is most influential in causing the model to make a particular prediction. For example, given the question "*who was the first U.S. president?*", the system should return the correct answer "*George Washington*" and references to training or retrieval corpora that are—to the model—causally linked to the answer. If the training or retrieval data included Washington's Wikipedia page, a typical human would expect for this to be included. However, the requirement we impose is causal and counterfactual: had the model not used that data, the prediction should change. If the prediction does not change, then from the causal perspective, there may be other data that is either more influential or duplicative (e.g., if `whitehouse.gov` is in the training data, it is duplicative). Next, we describe common semi-parametric models and sketch how this casually-based answer provenance could be obtained and computational challenges to overcome.

Provided an input prompt and retrieved text, semi-parametric models like ATLAS (Izacard et al., 2022) or passing documents as prompts to GPT-3 (Kasai et al., 2022) are adept at generating free-text, short answers. Likewise, parametric models with flexible input like GPT-3 can be combined with retrievers to achieve a similar goal; alternatively, transformer models can be retrofitted with layers so that passages can be integrated in embedding space (Borgeaud et al., 2021). While retrieval-augmentation is no catch-all panacea to model hallucination, it does mitigate the problem (Shuster et al., 2021b). Additionally, models' explanations can make it easier to know when to trust models and when not to (Feng and Boyd-Graber, 2022).

In the case of QA models that take question plus retrieved text as input, there are several options. First, the model could provide several alternative answers which provide insight into the distribution of model outputs, rather than just a point estimate. Second, the model could provide a combination of feature-based explanations such as token saliency maps and the model's confidence in a correct answer (Wallace et al., 2019b). When combined, they can jointly influence the degree to which humans trust the model (Lai and Tan, 2019). However, to provide a complete account of model behavior, we must return to the training of model and the data used. In short, we endeavor to identify the combination of input, training data, and retrieved text that caused the model to produce the distribution of outputs (i.e., answer(s)). This is, of course, challenging due to scale of language model training data like C4 (Raffel et al., 2020) and the Pile (Gao et al., 2020) and that establishing causal—and therefore more faithful—explanations of model behavior is difficult. Training data attribution is one promising idea in this direction—it uses gradient and embedding based methods to attribute inference behavior to training data (Akyürek et al., 2022). For example, influence functions (Hampel, 1974; Han et al., 2020) and TracIn (Pruthi et al., 2020) link predictions to specific training examples, but are

computationally expensive and are approximate rather than exact solutions. To firmly establish a causal connection, one could fully re-train the model without the identified training examples, but this is prohibitively expensive in practice. Future development of efficient training data attribution, combined with methods like interpretations of input plus retrieved data, is a promising direction towards more complete explanations of model predictions.

6.2 Tabular Engine

As described at the end of Section 4, the knowledge retriever will pass on the data obtained to PAG. The QAP module will pass information about its plan to PAG. If the data obtained is tabular and a SQL query is generated, the information is passed to the tabular engine of PAG to compute the required answer(s). The recent advances in Text-to-SQL (Wang et al., 2020; Zhao et al., 2022) provide a good technical foundation for generating such SQL queries.

In most cases, it is not difficult to understand the correspondence between the natural language question and the SQL query that is generated. Once the SQL query is obtained, provenance can be systematically derived. In databases, the notion of provenance is well-studied (Cheney et al., 2009) for a large class of SQL queries; from explaining why a tuple is in the output (i.e., the set of tuples in the database that led to the answer), where a value in a tuple is copied from (i.e., which cell in the source table is the value copied from) (Buneman et al., 2001) to how that tuple was derived, which is formalized as semirings (Green et al., 2007), a polynomial that essentially describes conjunction/disjunction of records required materialize a record in the result. Database provenance has also been extended to aggregate queries (Amsterdamer et al., 2011). Since one can derive the mapping between the input question and the SQL query that is generated and also derive the provenance from the data sources based on the SQL query, it becomes possible to understand how the input question led to the answers given by PostText.

Putting all together, PostText first explains that the name of the image (i.e., "*a good version of this dish*") referred in question is Shaking beef. It then shows the SQL query that is generated for the question "*Where can I find a good version of Shaking beef*" and the ranking function used for ranking the rows of restaurants with reviews for the dish Shaking beef. For our running example, the answer is obtained from the first row of the table in Figure 1. Specifically, the answer is summarized from the column *Dish* and *Review snippets/embeddings*. The actual snippets are found following the provenance links captured in the column *Provenance*. A more direct relationship between the summary and the actual review snippets can also be established (Carmeli et al., 2021).

The success of this approach depends on how far we can push database provenance systematically as SQL queries can still be far more complex than what is investigated in past research (e.g., complex arithmetic and aggregate functions involving also negation, group filters, and functions over values of different modalities). As an alternative to executing the SQL query over the tables obtained, the tabular engine can also choose to deploy table question answering (tableQA) methods where a model directly searches the tabular data for answers based on the input question (Sun et al., 2016). Tapas (Herzig et al., 2020) and Tapex (Liu et al., 2022) are two example solutions for tableQA that formulates tableQA as sequence understanding/generation tasks. Like other recent tableQA works (Glass et al., 2021; Herzig et al., 2021), they consider the problem of computing the answer from a single input. It will be interesting to explore how to explain the results obtained using tableQA methods and how tableQA methods can be extended to handle multi-hop questions where the answer may span multiple tables or involve different types of aggregations, reasoning and modalities.

7 Preliminary Findings

To test our hypothesis that views are valuable for answering queries, especially queries that involve counting or aggregation, we have implemented a first version of PostText[2] and compared it against some QA baselines.

The current implementation of PostText assumes views over the underlying data are available in tabular format. The QAP module simply routes the query to a view-based engine (VBE) or a retrieval-based engine (RBE) to answer the query. VBE picks the best view and translates the natural language query into an SQLite query against the view using OpenAI's `gpt-3.5-turbo/gpt-4` model. It then executes the SQLite query against

[2]PostText source code will be made available soon.

	VBE	RBE	DBChain	DBChain (no views)
S	3.45	2.81	3.37	2.72
M	3.79	2.69	3.28	2.61
L	3.11	2.44	2.95	1.95

Table 2: Results with GPT-3.5-turbo. Sizes of (S)mall, (M)edium, (L)arge are 1.1MB, 2.4MB, and 5.6MB respectively.

	VBE	RBE	DBChain	DBChain (no views)
S	3.33*	2.10*	2.14*	1.10*
M	3.55	1.93	2.35	1.51*
L	3.08	2	1.97	1.11*

Table 3: Results with GPT-4. * indicates that timeouts or API errors were encountered during experimentation.

the view to obtain a table result which is then translated into English as the final answer. VBE also analyzes the SQLite query to compute the provenance of the answers. At present, it does so by simply retrieving all tuples that contributed to every (nested) aggregated query that is a simple (select-from-where-groupby-having clause) and does not handle negations. An example of the VBE process is described in Appendix B. RBE is implemented with Langchain's `RetrievalQAwithSources` library. It first retrieves top-k documents that are relevant for the query and then conditions its answer based on the retrieval. The answer and the ids of the retrieved documents are returned.

For our experiments, we use the 42 multihop queries over 3 synthetic personal timelines of different sizes from TimelineQA's benchmark (Tan et al., 2023). The personal timelines model the daily activities (e.g., the trips made, things bought, people talked to) of a person over a period of time. We create a view around each type of activity (e.g., trips, shopping, daily_chats) for VBE. For further comparison, we also ran Langchain's SQLDatabaseChain (DBChain) to perform QA over the same VBE views. Furthermore, we ran it over timelines loosely structured as a binary relation of (date,description) pairs (called DBChain (no views)). We compared the returned answers against the ground truth answers by grading them on a scale of 1-5, with a LLM, where 5 means the returned answer has the same meaning as the ground truth answer (the grading scheme is described in the Appendix C).

Our results are shown in Tables 2 and 3. Across both tables, the results on DBChain vs. DBChain(no views) reveal that adding some structure (in this case adding views) is crucial for better performance. Although the benchmark is a relatively small dataset, the scale of the timelines already reveals an impact on the accuracy across all QA systems. For DBChain, the drop in accuracy as the size increases because it sometimes relies on generating SQL queries that return all relevant records and passing all the records to the language model to compute the aggregate. When the results returned are large, which tends to be the case for larger timelines, the token limit of the LLM is often exceeded. VBE has a similar downward trend. It tends to generate queries that push the aggregates to the SQL engine and hence, avoids the issue of exceeding the token limit of the language models for many cases encountered in DBChain. Still, as the timeline gets larger, the result returned by the generated SQL query tends to be bigger and when these results are passed to the verbalization component to compose an answer in English, this may sometimes exceed the token limit of the language model. We also found that on a handful of cases, it so happens that the SQL query generated for L is invalid compared with those generated for the sparse dataset.

The scores of RBE is relatively stable across all data densities. But overall, it tends to score lower compared with VBE and DBChain . This is because RBE relies on retrieving the top k documents from an index to condition the answers upon, regardless of the size of the timeline. However, these retrieved documents may not contain all the necessary information for answering the question in general. Even though the grading scores may not reveal this, the answers tend to be "more wrong" for aggregate queries over a larger timeline.

8 Conclusion

POSTTEXT enhances the core ideas of semi-parametric architectures with views, a query analyzer & planner, and a provenance-aware answer generator. Our initial results indicate that POSTTEXT is more effective on queries involving counting/aggregation when we provide structured views to facilitate computation. We plan to further develop and investigate POSTTEXT to automatically determine what views to construct, how does one generate plans and compare amongst plans, and how can one measure the quality of answers with provenance.

Limitations and Ethical Considerations

We point out the limitations of large language models (costly to train, deploy, maintain, hallucinate, opaque). The vision of POSTTEXT shows promise of less costly training, maintenance, and more explainability. However, no actual system is built yet to validate these claims and it is also not clear that a system with POSTTEXT architecture will be easier to deploy since it has more components.

References

Armen Aghajanyan, Bernie Huang, Candace Ross, Vladimir Karpukhin, Hu Xu, Naman Goyal, Dmytro Okhonko, Mandar Joshi, Gargi Ghosh, Mike Lewis, and Luke Zettlemoyer. 2022. CM3: A causal masked multimodal model of the internet. *CoRR*, abs/2201.07520.

Sanjay Agrawal, Surajit Chaudhuri, and Vivek R. Narasayya. 2000. Automated selection of materialized views and indexes in SQL databases. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 496–505. Morgan Kaufmann.

Sanjay Agrawal, Eric Chu, and Vivek Narasayya. 2006. Automatic physical design tuning: Workload as a sequence. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, page 683–694, New York, NY, USA. Association for Computing Machinery.

Ekin Akyürek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. 2022. Tracing knowledge in language models back to the training data. In *Findings of the Association for Computational Linguistics: EMNLP*. Association for Computational Linguistics.

Yael Amsterdamer, Daniel Deutch, and Val Tannen. 2011. Provenance for aggregate queries. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 153–164. ACM.

Simran Arora, Patrick Lewis, Angela Fan, Jacob Kahn, and Christopher Ré. 2022. Reasoning over public and private data in retrieval-based systems.

Akari Asai, Matt Gardner, and Hannaneh Hajishirzi. 2022. Evidentiality-guided generation for Knowledge-Intensive NLP tasks. In *Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy S. Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of Empirical Methods in Natural Language Processing*.

E. Bertino and R. Sandhu. 2005. Database security - concepts, approaches, and challenges. *IEEE Transactions on Dependable and Secure Computing*, 2(1):2–19.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W Rae, Erich Elsen, and Laurent Sifre. 2021. Improving language models by retrieving from trillions of tokens.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Nicolas Bruno and Surajit Chaudhuri. 2008. Constrained physical design tuning. *Proc. VLDB Endow.*, 1(1):4–15.

Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. 2001. Why and where: A characterization of data provenance. In *ICDT*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330.

Nofar Carmeli, Xiaolan Wang, Yoshihiko Suhara, Stefanos Angelidis, Yuliang Li, Jinfeng Li, and Wang-Chiew Tan. 2021. Constructing explainable opinion graphs from reviews. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia*, pages 3419–3431. ACM / IW3C2.

ChatGPT3-OpenAI. Chatgpt: Optimizing language models for dialogue.

Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2020. MOCHA: A dataset for training and evaluating generative reading comprehension metrics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6521–6532, Online. Association for Computational Linguistics.

Wenhu Chen, Hexiang Hu, Xi Chen, Pat Verga, and William W. Cohen. 2022. Murag: Multimodal retrieval-augmented generator for open question answering over images and text.

James Cheney, Laura Chiticariu, and Wang Chiew Tan. 2009. Provenance in databases: Why, how, and where. *Found. Trends Databases*, 1(4):379–474.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2022. Binding language models in symbolic languages.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Sudipto Das, Miroslav Grbic, Igor Ilic, Isidora Jovandic, Andrija Jovanovic, Vivek R. Narasayya, Miodrag Radulovic, Maja Stikic, Gaoxiang Xu, and Surajit Chaudhuri. 2019. Automatically indexing millions of databases in microsoft azure sql database. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, page 666–679, New York, NY, USA. Association for Computing Machinery.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41:391–407.

Emily Dinan, Gavin Abercrombie, A Stevie Bergman, Shannon Spruit, Dirk Hovy, Y-Lan Boureau, and Verena Rieser. 2021. Anticipating safety issues in E2E conversational AI: Framework and tooling.

Jesse Dunietz, Gregory Burnham, Akash Bharadwaj, Owen Rambow, Jennifer Chu-Carroll, and David Ferrucci. 2020. To test machine comprehension, start by defining comprehension. In *Proceedings of the Association for Computational Linguistics*.

Shi Feng and Jordan Boyd-Graber. 2022. Learning to explain selectively: A case study on question answering. In *Proceedings of Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800GB dataset of diverse text for language modeling.

Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. 2008. *Database Systems: The Complete Book*, 2 edition. Prentice Hall Press, USA.

Michael R. Glass, Mustafa Canim, Alfio Gliozzo, Saneem A. Chemmengath, Vishwajeet Kumar, Rishav Chakravarti, Avi Sil, Feifei Pan, Samarth Bharadwaj, and Nicolas Rodolfo Fauceglia. 2021. Capturing row and column semantics in transformer based question answering over tables. In *NAACL-HLT*, pages 1212–1224. Association for Computational Linguistics.

Jonathan Goldstein and Per-Åke Larson. 2001. Optimizing queries using materialized views: A practical, scalable solution. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, page 331–342, New York, NY, USA. Association for Computing Machinery.

Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. 2016. Deep image retrieval: Learning global representations for image search. In *Computer Vision – ECCV 2016*, pages 241–257, Cham. Springer International Publishing.

GPT-Index. 2022. [link].

Goetz Graefe. 1993. Query evaluation techniques for large databases. *ACM Comput. Surv.*, 25(2):73–169.

Todd J. Green, Gregory Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 31–40. ACM.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-Augmented language model Pre-Training. In *Proceedings of the International Conference of Machine Learning*.

Alon Y. Halevy. 2001. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294.

Frank R Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. In *NAACL-HLT*, pages 512–519. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *ACL*, pages 4320–4333. Association for Computational Linguistics.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal large language models.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, page 2333–2338, New York, NY, USA. Association for Computing Machinery.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Atlas: Few-shot learning with retrieval augmented language models.

Alekh Jindal, Konstantinos Karanasos, Sriram Rao, and Hiren Patel. 2018. Selecting subexpressions to materialize at datacenter scale. *Proc. VLDB Endow.*, 11(7):800–812.

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5684–5696. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. 2022. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, and Kentaro Inui. 2022. RealTime QA: What's the answer right now? *arXiv [cs.CL]*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA. Association for Computational Linguistics.

Vivian Lai and Chenhao Tan. 2019. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. Association for Computing Machinery.

LangChain. [link].

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of Advances in Neural Information Processing Systems*.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: table pre-training via learning a neural SQL executor. In *ICLR*. OpenReview.net.

Thomas Macaulay. 2020. Someone let a gpt-3 bot loose on reddit — it didn't end well.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5783–5797, Online. Association for Computational Linguistics.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. UniK-QA: Unified representations of structured and unstructured knowledge for open-domain question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1535–1546, Seattle, United States. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022.

Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. In *Proceedings of Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. RASAT: Integrating relational structures into pretrained seq2seq model for text-to-sql. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified Text-to-Text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Karl Schnaitter, Serge Abiteboul, Tova Milo, and Neoklis Polyzotis. 2007. On-line index selection for shifting workloads. In *2007 IEEE 23rd International Conference on Data Engineering Workshop*, pages 459–468.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shelly Sheynin, Oron Ashual, Adam Polyak, Uriel Singer, Oran Gafni, Eliya Nachmani, and Yaniv Taigman. 2022. Knn-diffusion: Image generation via large-scale retrieval.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021a. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 3784–3803. Association for Computational Linguistics.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021b. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP*. Association for Computational Linguistics.

Chenglei Si, Chen Zhao, and Jordan Boyd-Graber. 2021. What's in a name? answer equivalence for open-domain question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9623–9629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chenglei Si, Chen Zhao, Sewon Min, and Jordan L. Boyd-Graber. 2022. Revisiting calibration for question answering. *ArXiv*, abs/2205.12507.

Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 771–782, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Wang-Chiew Tan, Jane Dwivedi-Yu, Yuliang Li, Lambert Mathias, Marzieh Saeidi, Jing Nathan Yan, and Alon Y. Halevy. 2023. Timelineqa: A benchmark for question answering over timelines. In *ACL (to appear)*.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019a. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Eric Wallace, Pedro Rodriguez, Shi Feng, and Jordan Boyd-Graber. 2019b. Trick me if you can: Human-in-the-loop generation of adversarial question answering examples. In *Transactions of the Association for Computational Linguistics*, pages 387–401.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: relation-aware schema encoding and linking for text-to-sql parsers. In *ACL*, pages 7567–7578. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan

Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick S. H. Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models.

Michihiro Yasunaga, Armen Aghajanyan, Weijia Shi, Rich James, Jure Leskovec, Percy Liang, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2022a. Retrieval-augmented multimodal language modeling.

Michihiro Yasunaga, Armen Aghajanyan, Weijia Shi, Rich James, Jure Leskovec, Percy Liang, Mike Lewis, Luke Zettlemoyer, and Wen-Tau Yih. 2022b. Retrieval-Augmented multimodal language modeling.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. 2022. Socratic models: Composing zero-shot multimodal reasoning with language.

Chen Zhao, Yu Su, Adam Pauls, and Emmanouil Antonios Platanios. 2022. Bridging the generalization gap in text-to-sql parsing with schema expansion. In *ACL (1)*, pages 5568–5578. Association for Computational Linguistics.

Chen Zhao, Chenyan Xiong, Hal Daumé III, and Jordan Boyd-Graber. 2021. Multi-step reasoning over unstructured text with beam dense retrieval. In *North American Association of Computational Linguistics*.

A Appendix

B View-based QA

Example run of POSTTEXT with the query "*When was the last time I chatted with Avery?*":

This query is first matched against a set of available views and the best one is picked if there is sufficient confidence. In this case, the view `daily_chat_log` is selected.

The query is first translated into an SQLite query:

```
SELECT MAX(date)
FROM daily_chat_log
WHERE friends LIKE '%Avery%'
```

The SQLite query is then cleaned and "relaxed". For example, on occasions, an attribute that does not exist is used in the query even though this happens rarely. In this case, no cleaning is required. The conditions over TEXT types are also relaxed. We convert equality conditions (e.g., `friends = 'Avery'`) to `LIKE` conditions (e.g., `friends LIKE '%Avery%'`) and further relax LIKE condition with a user-defined `CLOSE_ENOUGH` predicate.

```
SELECT MAX(date)
FROM daily_chat_log
WHERE (friends LIKE '%Avery%' OR
       CLOSE_ENOUGH('%Avery%', friends))
```

The above query is executed and the results obtained is shown below. We then verbalized an answer based on the table result. **Result:** [('2022/12/26')]

Returned answer (verbalized): *The last time I chatted with Avery was on December 26, 2022.*

We observe that Langchain's SQLDatabaseChain provides a very similar functionality of matching an incoming query against available tables and generating an SQL query over

the matched tables. However, SQLDatabaseChain does not clean or relax query predicates, and requires one to specify a limit on the number of records returned. Furthermore, it does not compute the provenance of the answer obtained, as we will describe in the next section. As we also described in Section 7, view-based QA generally outperforms SQLDatabaseChain because of its ability to push aggregates to the database engine instead of relying on the language model to aggregate the results (after using the database engine to compute the relevant records for answering the query.

Provenance queries: PostText generates queries to retrieve records that contributed to the answer returned above. It does so by analyzing every `select-from-where-groupby-having` subquery in the generated query to find tuples that contributed to every such subquery. For example, the following SQL queries are generated to compute provenance.

```
SELECT name
FROM pragma_table_info('daily_chat_log')
where pk;

q0:
SELECT eid
FROM daily_chat_log
WHERE (friends LIKE '%Avery%' OR
       CLOSE_ENOUGH('%Avery%', friends))
```

The first query above returns the key of the table and the second retrieves the keys from the table that contributed to the returned answer.

```
[('q0', ('e152',)), ('q0', ('e154',)), ('q0',
  ('e169',)), ('q0', ('e176',)), ...]
```

C Grading scheme

The following is our grading scheme used for grading the answers generated by different systems against the ground truth answer:

- 5 means the systems's answer has the same meaning as the TRUE answer.
- 4 means the TRUE answer can be determined from the system's answer.
- 3 means there is some overlap in the system's answer and the TRUE answer.
- means there is little overlap in the system's answer and the TRUE answer.
- 1 means the system's answer is wrong, it has no relationship with the TRUE answer.