ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA241052

Constrained LLM-Based Query Generation for Question Answering on Official Statistics

Jonas Kouwenhoven [a], **Lucas Lageweg** [b] **and Benno Kruit** [c]

[a]jonaskouwenhoven@live.nl
[b]l.lageweg@cbs.nl
[c]b.b.kruit@vu.nl

Abstract. This research explores the development of a knowledge graph statistical question answering system for Statistics Netherlands. Aimed at efficiently retrieving single statistical values from their extensive database, which encompasses over a billion values across more than 4,000 tables, we propose a comprehensive three-component framework consisting of: (1) a data augmentation method to generate synthetic data, (2) an entity retrieval system that leverages various encoder networks along with different hard negative mining techniques for the effective retrieval of tables, measures, and dimensions, and (3) an innovative large language model-based query generator. A central innovation of our research is the introduction of a dynamic prompting technique for query generation, which creates prompts specifically for a certain phase of the token generation. This approach ensures that the model is supplied with information relevant for generating specific tokens in a symbolic query. With this approach, we propose a novel system that can help find relevant information in official statistics and similar systems, which is vital for governmental decision making and all fields of research utilising and relying on these statistics.

1 Introduction

Statistics Netherlands (Centraal Bureau voor de Statistiek; CBS) is an independent administrative body of the Dutch government tasked with the creation of statistics over a broad spectrum of social topics, and the responsibility to make them accessible to the general public. However, in-house studies have shown that users struggle to find the correct tables for their needs in the vast amount of data available. This research aims to develop a Question Answering (QA) system to provide specific statistical observations from this data as responses to natural-language user questions.

QA systems can take several forms, with most recently free-form generative Large Language Models (LLMs) like ChatGPT [18] and GPT4 [19] getting much attention. Due to the nature of these models, they are able to generalize very well on a large range of topics, but have shown to be prone to 'hallucinations', where plausible but incorrect or even nonsensical answers are generated [27]. Especially for official data like governmental statistics, this is highly undesirable behaviour.

Knowledge Graph Question Answering (KGQA) is a field where knowledge graphs (KGs) containing real-world facts and relations in structured form are used as a basis for QA systems. Answers of such systems should always adhere to the KG. Therefore, assuming it contains correct information, answering by returning parts of the KG, or reasoning over it, cannot lead to nonsensical answers. In this paper, we introduce an end-to-end pipeline for a generation-based KGQA system of CBS data. Our approach introduces a data augmentation process for enhancing model training, explores various encoder architectures for entity retrieval and proposes a new query generator mechanism enhanced by Low Rank Adaptation (LoRA). Additionally, we propose a new prompting technique that utilizes dynamic prompts, constructing specific prompts based on the generation phase. These improvements help the process of generating symbolic expressions for querying a KG, and thereby enhancing the overall performance of the QA system.

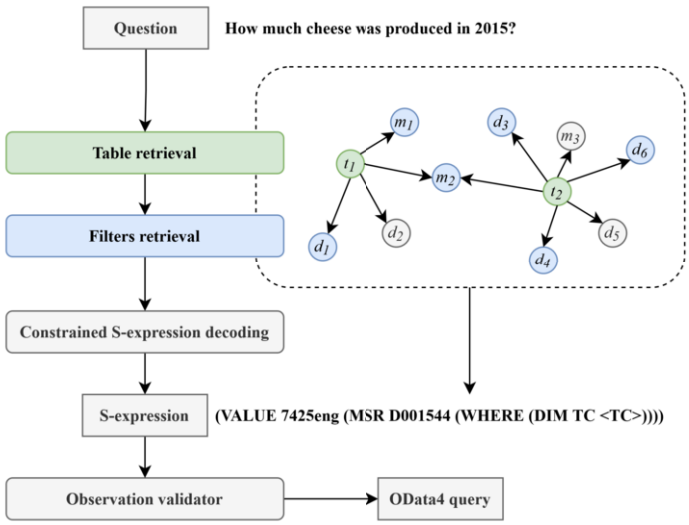
In this research we focus on the retrieval of a single table cell based on a given input query. In practice, this means that the system is able to answer questions that have an answer in a single cell of a table that is available in the KG. For our investigation, we assume only Dutch input questions, and focus on the top 500 most consulted tables on the CBS website. In short, our contributions are as follows:

- a data augmentation method for diverse, realistic question generation that can be used for training (Sections 3.3 & 4.1);
- the adaptation and evaluation of several state of the art approaches for entity retrieval in official statistics (Sections 3.4 & 5.1);
- a novel method for query generation based on low-rank adaptation of LLMs using dynamic prompting (Section 3.6.2);
- improvements upon strong QA-baselines, on both in-domain and out-of-domain data (Section 5);
- we discuss the current state of deployment and potential future impact of the KGQA system for official statistics (Section 7).

2 Related Work

Query Generation Systems mostly related to our approach are query generation-based QA systems, in particular text-to-SQL. In their survey, Qin et al. [20] explain several approaches on learning input and table schema representations (encoding), in order to later generate and parse SQL statements (decoding).

Generative models are also considered for the task of generating logical forms to retrieve and reason over KGs themselves. However, adapting these models still provides a problem when it comes to producing queries/expressions that are faithful to the KG (i.e. to prevent querying non-existing triples). Current state-of-the-art methods for KGQA use KG grounding for constraining queries that adhere to the KG, as is shown by Gu et al. [4], Yu et al. [26] and Shu et al. [23]. All



(a) Overview

Periods	Dairy production	
	Butter	Cheese
	1 000 kg	
1995	132,300	682,900
2000	126,200	683,600
2005	118,800	672,200
2010	133,419	752,638
2015	147,577	844,974

(b) Example Table Fragment

Figure 1: (a) Overview of our pipeline from query to answer. Candidate table nodes (green) are retrieved for the query, after which the measure and dimension filter candidates are retrieved (blue), resulting in a complete subgraph for the table candidates. The subgraph is used as input for the constrained S-expression decoding by either the baseline method or trained model. (b) Example CBS table fragment (from `7425eng`), showing one dimension (time periods) and two measures.

three examples take roughly the same overall approach, and serve as the main source of inspiration for this current research.

In their work, Schneider et al. [22] investigate the performance of different LLMs for query generation in text-to-SPARQL tasks. They evaluated one- and few-shot prompting on multiple LLMs. They found that zero-shot prompting in combination with a trained LoRA on LLaMA2 performed best based on the different benchmarks. Nan et al. [13] propose prompt design techniques to enhance Text-to-SQL systems. Their approach leverages example SQL structures and integrates KGs, achieving significant performance boosts on the Spider dataset.

Data Augmentation Similar to the approach we implemented for data augmentation, Bonifacio et al. [1] introduced InPars, a methodology aimed at creating synthetic training datasets for Information Retrieval (IR) tasks. Utilizing a document and multiple pairs of questions as input, their technique leverages language models to generate a question that aligns with said document. They assessed the efficacy of this method by evaluating 10,000 synthetically generated questions with the GPT-3 Curie language model across various IR benchmarks. The results showcased a notable improvement over traditional unsupervised sparse methods like BM25 [1]. Jeronymo et al. [9] employed a reranker as a filtering mechanism to select only the best synthetically generated examples.

Entity Retriever The Entity Retriever (ER) in QA systems plays a critical role, selecting the first answer candidates that function as input for the rest of a QA pipeline. Methods for ER are broadly categorized into sparse and dense approaches [6]. While sparse methods rely on word co-occurrences, dense methods use high-dimensional vector spaces to capture the nuanced relationships between queries and documents. When using sparse methods, a challenge arises due to the terminology gap between CBS and its users. Non-frequent users often formulate queries using language dissimilar to CBS' jargon. For instance, a user might ask "How many buildings are in Amsterdam?", whereas CBS would phrase it as "Stock of immovable properties in Amsterdam". This difference in wording can lead to mismatches during exact word-based query-to-table comparisons. This research will therefore focus on dense retrieval methods as they tend to capture more contextual similarities.

Significant advancements in dense retrieval have been achieved through the development of architectures like DPR (Dense Passage Retrieval) and ColBERT [11, 12]. Both methods operate on a Bi-Encoder architecture, independently processing queries and documents before comparing their vector embeddings for similarity. This method has demonstrated substantial improvements over traditional retrieval models [11]. ColBERT, leveraging a late interaction model, offers refined term-level matching capabilities, enhancing the retrieval process further [12].

3 Methodology

This section outlines the methodology for the proposed end-to-end KGQA pipeline (see Figure 1). The end-to-end pipeline consists of four parts. The first step is *table retrieval* based on the query to determine the relevant tables (Section 3.4.1), which is followed by *filters retrieval*, which selects the relevant measures and dimension within the table (Section 3.4.2). Then, the retrieved entities (i.e., tables, measures and dimensions) are passed on to the query generator (Section 3.6), which utilizes the entities to formulate the final S-expression (Section 3.2). Finally the S-expression is converted into a OData4 query that retrieves the table cell deemed most relevant for the input query.

3.1 Public Data of Statistics Netherlands

For this research, to narrow the scope, we will use CBS data that is publicly available and, more specifically, will use the 500 most consulted tables on the CBS website. All data is made available via the ISO/IEC approved Open Data Version 4.01 protocol (OData4) [14, 8], which is the API that will be queried to return observations to user questions. Each table observation consists of a single measure value (i.e. a statistic being measured) and values for all dimensions

available in that table (i.e. filtering characteristics or properties for said measures). CBS maintains a public vocabulary of concepts[1], in which every unique measure and dimension has an identifier. In the editorial process of publishing statistics, all measures and dimensions are standardised as much as possible in an attempt to maintain consistency between tables. In this work, we make use of a subset of this vocabulary encoded as RDF in our KG, corresponding to the schema descriptions of the tables in our target sample. In Fig. 1b, a fragment of an example table is shown, with one dimension (time periods) and two measures).

3.2 Query Syntax: S-expressions

S-expressions are symbolic expressions which contain atoms and expressions (which are always S-expressions themselves) in a tree-like structure as nested lists. For our purpose, Gu et al. [4] propose a format where the expression comprises of functions (`AND`, `COUNT`, `JOIN`, etc.) and entities (i.e. the atoms). The S-expressions always denote operations over the KG. ArcaneQA [3] extends their definition with functions for general and temporal constraints but follows the same principle. The benefit of using S-expressions is that they are compact and concise, human-readable and machine-interpretable and, most importantly, easily converted to other types of querying formats like SPARQL or OData4. An expression will always start with an aggregation/operator function to indicate the operation needed to be done over the values denoted in the expression that follows. Two special atom placeholders exist for the time and geographical constraint functions (`TC` and `GC`).

3.3 Data Annotation

CBS has developed a method for manual data annotation. This method involves annotators writing queries that can be answered by a specific table cell. Annotators were instructed to write their questions both as full sentences and in a more casual style, aiming to simulate the formulation of questions posed by users in a search engine. The data obtained from this manual annotation process contains queries and their corresponding S-expression, resulting in 2300 annotated pairs.

The annotated queries were distributed over random tables from the CBS datapool, and contained a strong class imbalance towards tables that were more easily annotated. This class imbalance and random distribution motivates extending this study with data augmentation. In this extension, annotated S-expressions and their associated queries are used to fine-tune a GPT-3.5 model through the OpenAI fine-tuning services [17]. The query-expression pairs were transformed into prompts using the descriptions of the IDs for various measures, dimensions, and table IDs, as illustrated by Example 1. Training such a model reduces the need for additional manual annotation, while also significantly increasing the amount of annotated data.

Example 1. *Automatic data annotation*
Original S-expression:
(VALUE (83822NED (MSR M001191_2 (WHERE (DIM Bedrijfsgrootte WP19114) (DIM TC 2015JJ00)))))
Entities:
83822NED: Fitness Centers; customers, subscription prices, and facilities
M001191_2: Total Customers (x 1000)
WP19114: 0 to 3 employees
2015JJ00: 2015
Prompt Transformation:
Think of a question for the table 'Fitness Centers; customers, subscription prices, and facilities', with columns 'Total Customers (x 1000)' and rows '0 to 3 employees', '2015'
Query Generated by the Fine-tuned Model:
How many customers did a fitness center with 0 to 3 employees have in 2015?

[1] https://vocabs.cbs.nl/en/ (Accessed: 26-04-2024)

3.4 Entity Retrieval

The entity retriever component within the overall QA pipeline is tasked with identifying the most relevant entities (i.e., tables, measures, and dimensions) based on an input query. In this research, we split the entity retriever into two components. First, we aim to identify the most relevant tables (Section 3.4.1). Following that, we aim to identify the most relevant table filters (Section 3.4.2).

For table and filters retrieval, several baselines, both sparse and dense, and two Bi-Encoder network architectures were investigated: a simple Dual-Encoder architecture and the more sophisticated ColBERT model.

3.4.1 Table Retrieval

The first stage of the retriever focuses on fetching the most relevant tables given an input query. Both sparse and dense methods are evaluated for this task. Dense methods utilize pre-computed embeddings to calculate similarity scores. Sentence Transformers [21] can be used with a variety of pre-trained models that transform input queries into high-dimensional vectors. The final similarity score between a table and a query is then calculated using the cosine similarity of these vectors. Here, we used a Dutch language GroNLP BERT sentence transformer [2] for creating the embeddings. However, GroNLP has a significant limitation: a maximum token length of 75. While the queries in our datasets typically adhere to this length, the table summaries and descriptions that are embedded often exceed this limit.

Additionally, table titles lack descriptiveness, hindering the capture of table information through the title embeddings exclusively. To address this, an automated augmentation process was implemented using the OpenAI GPT-3.5-turbo model. A prompt was created using the table title, summary, description, and a random sample of its measures. The model was tasked to generate a summary of no more than 75 tokens based on the provided information. These generated summaries were used throughout the rest of the research.

Baselines For the entity retriever, three different baseline algorithms are considered: a sparse TF-IDF-based BM25-method, the GroNLP embeddings of the generated summaries, and the OpenAI `text-embedding-ada-002` model. As a Dutch governmental instutition, however, CBS is strictly advised to not send user data to any of the OpenAI services [16], and therefore this baseline is only used for reverence to performance of the models.

Dual-Encoder Following the approach by Karpukhin et al. [11], we implemented and evaluated a Dual-Encoder architecture. The architecture takes as input the GroNLP embedings of the generated table summaries and the query. The model uses the input structure $\langle a, p, n \rangle$, where a is the anchor or query embedding, p is the positive table embedding, and n represents a negative table embedding (i.e. a table that does not contain the answer to the query). The model aims to minimize the distance between a and p, while maximizing the distance between a and n. To achieve this, a triplet loss function was implemented [25]. To effectively train the Dual-Encoder we need

to sample multiple samples that challenge the network; instances that are similar to the query but do not contain the actual answer. This task of sampling is usually referred to as hard negative mining. In our research we use cosine similarity to select the top-k most and least similar tables (p and n respectively) relative to a, resulting in $2k$ negatives examples per query in the train set. This approach is similar to that presented by Smirnov et al. [24].

ColBERT Introduced by Khattab and Zaharia [12], ColBERT is a state-of-the-art retrieval model that generates embeddings for each term in a document separately. This allows for a more fine-grained matching between the query and the document compared to single vector representations. Instead of directly comparing single terms, ColBERT uses a late interaction approach. Here, it calculates a similarity score between a query and a document by finding how well the meaning of each query term aligns with the most relevant parts of the document. This could greatly improve the ability of the model to connect specific parts of queries to specific parts of table descriptions, and thereby capturing more nuanced relationsships. In our implementation, an adaptation to the original ColBERT was made, utilising the aforementioned GroNLP embeddings.

3.4.2 Filters Retrieval

As described in Section 3.1, a single table cell consists of at least one measure and one or more dimensions, i.e. the *table filters*. Doing this, entity retrieval can be decomposed into two subtasks: identifying the relevant measure within a table that corresponds to the user query (measure retrieval), and finding the appropriate dimension(s) associated with the retrieved table (dimension retrieval). For both sub-tasks we leverage a weighted average of BM25+ scores and cosine similarity on GroNLP embeddings of entity descriptions to retrieve the most relevant entities for each subtask.

3.5 Low-Rank Adaptation

While LLMs revolutionized the field of natural language processing, fine-tuning them for specific tasks becomes increasingly difficult due to the massive computational resources required for full fine-tuning. This has driven the development of alternative, more efficient fine-tuning approaches.

Hu et al. [5] propose a solution to the challenge of fine-tuning LLMs by freezing the pre-trained model weights and introducing a trainable low-rank matrix in each layer of the architecture. This technique, known as Low-Rank Adaptation (LoRA), significantly reduces the number of parameters that needs to be trained. In a traditional fine-tuning procedure, adjustments are made to all the model weights to suit a new task. This process involves modifying the original weight matrix W, denoted as ΔW, resulting in updated weights $W + \Delta W$. Instead of modifying the W matrix directly, the aim of LoRA is to decompose the ΔW matrix. LoRA achieves the decomposition of ΔW by representing it as the product of two smaller matrices, A and B. The updated weight matrix W' becomes: $W' = W + BA$. In this equation, W remains fixed, while A and B have lower dimensionality, with their product $B \cdot A$ representing a low-rank estimation of ΔW. By selecting matrices A and B to have a lower rank r, the number of trainable parameters is significantly reduced. To demonstrate the decrease in trainable parameters, consider a scenario where W represents a $d \times d$ matrix. Typically, updating W would require updating d^2 parameters. However, if both B and A are reduced to dimensions $d \times r$ and $r \times d$ respectively, the overall number of parameters to be trained reduces to $2dr$, which is a significant reduction when r is less than d.

3.6 Query Generation

3.6.1 Baseline

An initial rule-based greedy baseline is used to generate the aforementioned S-expressions. By creating an S-expressions token by token, the generation is constrained to ensure that at each step admissible tokens are retrieved based on a given subgraph generated from the retrieved candidate entities. The greedy baselines utilise the scores from the retrieval step (Section 3.4) in a deterministic selection process. The graph nodes with the highest similarity scores to the query are selected for use in the generated S-expression, resulting in the best-scoring table, measure, and dimensions to be selected.

3.6.2 Decoder-Only Approach (LLM-based)

The second method for constrained generation of S-expression is a novel method for using a decoder-only model architecture. In this method, the generation is contrained on a sequence of tokens instead of a single token as propossed in earlier encoder-decoder-based models like ArcaneQA [3]. When embedding the graph identifiers, occuring in generated prompts and S-expressions, all identifiers need to be added to the models' tokenizer vocabularies, and their embeddings must be pre-trained to be able to embed their contextualized meaning (i.e. without any extra context, the model will not be able to create a relevant embedding for the identifier `7425eng` by default). This results in scalability issues when using a larger KGs, as is the case here, sometimes leading to the doubling of its vocabulary. In this decoder-only approach, however, when generating a token at a given timestep, the model evaluates the sequences in the list of admissible/constrained tokens and selects the sequence with the highest assigned score. This process is described by the following formula:

$$Y_{i,\text{next sequence}} = \underset{S \in \mathcal{S}}{\operatorname{argmax}} \left(\sum_{y \in S} \log P(y|X, Y_i) \right)$$

In this formula, $Y_{\text{next sequence}}$ represents the next sequence of tokens to be selected from a set of possible (sub)tokens. The selection is made by identifying the sequence S that maximizes the log probability of its tokens. Here, $\sum_{y \in S} \log P(y|X, Y_i)$ computes the total log probability of the next tokens y within the set of sequences $\mathcal{S}$, given the input prompt X and the current output sequence up to the i-th position, denoted by Y_i. For example, when `7425eng` is given to the decoder as one of the admissible next tokens, but only a decomposition of sub-tokens can be embedded by the model (e.g. `7425` followed by `##eng`), the summed log probability for these subtokens will determine the total probability of selecting this identifier for generation.

Dynamic prompting The novelty in this constraining method is the introduction of *dynamic prompting*, which, instead of calculating the likelihood of a token sequence based on a static prompt, adjusts the prompts according to the generation phase. For example, when generating a table ID, the prompt is altered to only include the most relevant table IDs and their descriptions. Similarly, when measures are generated in the next phase, it retrieves the measures related to the previously generated table ID, and using those to construct a new prompt. This method applies to the different dimension groups as well. Figure 2 contains a schematic overview of the dynamic prompting

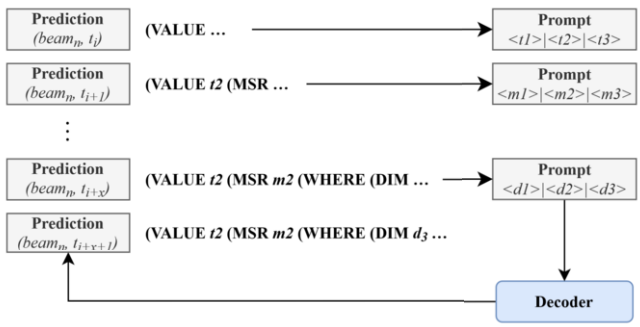


Figure 2: Schematic overview of dynamic promting for a decoder-only model architecture. If there are multiple tokens that can be generated, a custom prompt is given to the decoder with labels for all the possible options, as detailed in Section 3.6.2.

technique. This approach tackles the scalability issues compared to encoder-decoder methods. Static prompts, when including all relevant tables, measures, and dimensions, quickly reach the context limit of the current LLMs considered. Dynamic prompting, however, allows for a much larger initial set of tables, measures, and dimensions by focusing on the information relevant to the current generation step. This maximizes the efficiency of the LLM's context length. This technique could theoretically be applied to any open-source large language model (LLM) without fine-tuning. However, computational limitations restrict the research to models with 7 billion parameters. Due to the limited representation capabilities of these models and the success of LoRA in query generation as shown in Schneider et al. [22], the decision was made to train a LoRA model (See Section 3.5) on this task using dynamic prompting. The training involved creating a dynamic dataset mirroring the prompts in the final generation phase.

3.6.3 *Observation Validation*

The observation validator converts generated S-expressions into OData4 queries, incorporating several validation steps. Initially, it verifies that all necessary dimension groups are specified in the S-expression (i.e. a value is specified for all possible filters of the corresponding table of the S-expression). Missing dimensions are identified using the table subgraph. For missing dimensions, if applicable, default assumptions are made:

TimeDimension - the latest period is assumed;
GeoDimension - the largest geographical aggregate (usually the Netherlands) is assumed;
Other - a dimension denoting a total is assumed (if available).

If assumptions cannot be made, the validation fails, prompting the user to specify the missing information. The final step involves executing the OData4 query to retrieve observations and formulating the final answer.

4 Experimental Setup

4.1 *Benchmark Datasets*

Leveraging the ability to generate synthetic annotated data, as detailed in Section 3.3, a total number of 18686 queries were obtained, from which 2296 instances were manually annotated, and 16390 generated using the finetuned GPT-3.5 model. Using these queries the research was conducted using two datasets:

1. **T500**: a larger dataset comprising question-answer pairs for the top 500 most consulted tables on the CBS website.
2. Out-of-domain (**OOD**) dataset: manually annotated instances that are not present within the T500 dataset. An additional filtering mechanism was developed as detailed in Section 4.2 to filter out any semi-in domain instances for this dataset. This dataset is used to evaluate the model's performance on generalizing to unseen tables, and contains 203 question-answer pairs from 24 tables.

Diversity Index For the T500 dataset, a deliberate class imbalance is implemented. This imbalance is based on the idea that tables capable of answering a more diverse array of questions require more samples in the training data. To evaluate the diversity of a table, the number of unique terms in the description of the measures and dimensions of a table is plotted against the total number of values (observations), as shown in Figure 3. The diversity index relies unique terms instead of the total number of dimensions and measure combinations for a table, as some dimensions such as age ranges (e.g. *15 to 25 years old*, *25 to 35 years old*) are very similar and do not contribute significantly to question diversity. After obtaining the number of unique terms across all measures and dimensions in a table and its number of unique observations, the diversity index is calculated by dividing the number of terms by the number of observations. All tables are then sorted into five classes representing the percentile in which their diversity index falls. If a table is in class 5, it means it belongs to the class with the most diverse tables. When sampling queries for the train-test split, this diversity index is then multiplied by 5 to obtain the final number of queries in the train-test split for a table. This diverse sampling technique creates a final training dataset of 5,988 queries and a test set of 1,497 queries, omitting a significant portion of the data for training the entity retrievers.

4.2 *Cleaning the Out-of-Domain dataset*

Originally, OOD dataset was created by using queries that were manually annotated but did not belong to any of the tables in the T500 dataset. However, this method results in a limitation. Some tables in the constructed OOD dataset are near-duplicates of tables in the T500 dataset, making them in-domain rather than truly out-of-domain questions. For example, the table with the title: "Population 15 to 75 years; education level, districts and neighbourhoods, *2013*", compared to "Population 15 to 75 years; education level, districts and neighbourhoods, *2021*" present in the OOD and T500 datasets respectively. Therefore, this example cannot be considered an OOD table instance. To overcome this limitation, a cosine similarity is applied to the GroNLP embeddings of the table titles in the OOD and T500 datasets. If an OOD table instance had a similarity score higher than 75% compared to a table in the T500 dataset, it is omitted from the dataset. This filtering procedure ensured that all instances in the OOD dataset can truly be considered out-of-domain, rather than semi in-domain data.

4.3 *Query Generator*

From the entity retrieval step, the query generator obtains the 10 most relevant tables. From each of these tables, it also obtains the five most relevant measures. Finally, a total of 15 dimension IDs are obtained, spread over the different dimension groups. For instance, if a table has 3 different dimension groups, the 5 most relevant dimension IDs for each dimension group are obtained.

The query generator's performance is evaluated using various retrieval configurations. To demonstrate the effectiveness of Low-Rank

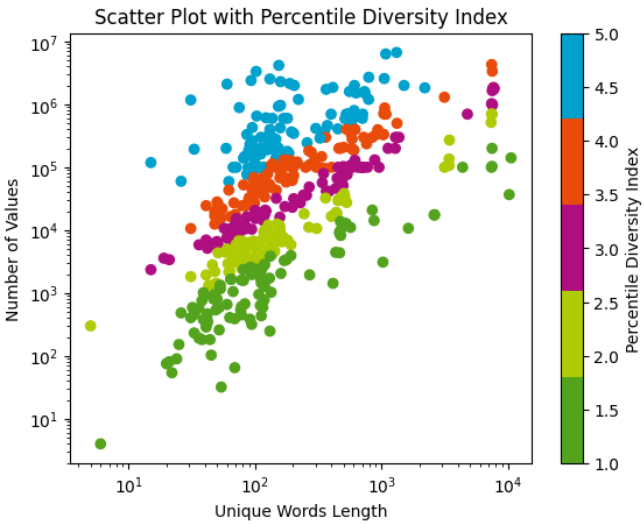


Figure 3: The diversity index of the the T500 dataset plotted with percentile class distribution labels colored. Each point is one table t, with its number of values v on the y-axis and the number of unique words w in the table and its filter descriptions on the x-axis, and the diversity index is calculated as $D_t = w/v$. Additional benchmark questions are included in the training set for tables containing more words per value (Section 4.1).

Adaptation (LoRA) training, the model is assessed with and without a trained LoRA. The employed LoRA is trained on a quantized Mistral-7B model [10]. The specific training configuration uses a LoRA rank of 16, the Adam optimizer for 80 epochs, and a learning rate of 0.0002.

4.4 Evaluation Metrics

For the table and measure codes within the generated S-expressions, accuracy (**Acc**) is used as a metric since only one of each type can exist in a valid expression. Dimension IDs, on the other hand, can appear multiple times, so the harmonic mean for the recall and precision of the dimensions is calculated (**F1**). A relevancy score (**RS**) is assessed using an scoring method leveraging GPT-3.5-turbo to evaluate the generated S-expression based on the input query with a score ranging from 0 to 1. This accounts for the possibility of multiple table cells corresponding to a single query. To determine the corectness of these generated relevancy scores, 100 answers were annotated manually and compared with the GPT-generated scores, yielding an average scoring difference of -0.067 (indicating the GPT model considers answers slightly more relevant than the annotators); this we consider acceptable for our purposes of determining the RS metric.

5 Results and Discussion

5.1 Entity Retriever

Table 1 showcases the performance of various baselines and Bi-Encoder models on the table retrieval task. As indicated in the table, both Bi-Encoder architectures achieve the highest top-k recall for different values of k. Specifically, they both yield a top-1 accuracy of 0.6. The top-performing baseline algorithm is the one that employs cosine similarity on the embeddings produced by the OpenAI-ada model. However, it is important to note that CBS is strongly advised against using OpenAI models in a pipeline environment. Therefore, this model serves solely for demonstration purposes, as it is considered state-of-the-art for generating embeddings. With the introduction of our proposed Bi-Encoder architectures, we have managed to outperform this state-of-the-art model. This allows CBS to eliminate the need for the OpenAI model but also provides an improved, smaller and overall more simplistic retrieval model.

Entity Retriever	Top 1	Top 5	Top 10
BM25+	0.44	0.74	0.83
GroNLP Cosine	0.36	0.64	0.74
ColBERT	**0.60**	0.88	**0.94**
Dual-Encoder	**0.60**	**0.90**	**0.94**
OpenAI-ada Cosine	*0.53*	*0.83*	*0.91*

Table 1: The Top-k recall of table retrieval for the various encoders and baselines on the T500 dataset.

5.2 T500 Benchmark

Table 2 presents the performance of the proposed LLM-based query generator and baseline models on the T500 dataset. The table presents various configurations of the table retriever and the query generator. The approach to filters retrieval, including measure and dimension retrieval, is consistent across all configurations, as is detailed in Section 3.4.2. As can be seen in the table, the best results are achieved by combining the proposed model solution with either of the Bi-Encoders. Specifically, it yields a table accuracy of 0.61 with the DualEncoder and 0.63 for ColBERT. The results indicate the effectiveness of training LoRA parameters for this task. When comparing models with and without LoRA, we see a performance boost with LoRA parameters. For example, ColBERT's table accuracy increased from 0.58 to 0.63. This improvement is even more prominent when examining the Dual-Encoder with the Mistral configurations; the implementation of LoRA parameters boosted table accuracy from 0.40 to 0.61. The relevance score (RS) indicates that the models can retrieve relevant table cells to input queries. This suggests that a single query can in many cases be answered by multiple table cells, and that relying solely on the accuracy and F1 metrics may not capture the model's capabilities and overall performance.

Table Retriever	Query Generator	TAB Acc	MSR Acc	DIM F1	RS
BM25+	Baseline	0.45	0.31	0.41	0.63
GroNLP Cosine		0.33	0.24	0.35	0.59
Dual-Encoder		0.55	0.35	0.46	0.65
ColBERT		0.58	0.39	0.51	0.67
OpenAI-ada Cosine		*0.51*	*0.35*	*0.45*	0.64
Dual-Encoder	Mistral - 7B (LoRA)	0.61	**0.42**	**0.53**	0.69
ColBERT		**0.63**	0.41	0.51	**0.71**
BM25+		0.51	0.34	0.45	0.67
Dual-Encoder	Mistral - 7B	0.40	0.25	0.37	0.61

Table 2: The results for the different Table Retrievers and Query Generators on the test set of the T500 dataset. The Mistral model in combination with the Dual-Encoder yield the highest results.

5.3 Out-of-Domain Benchmark

The results on the OOD dataset are presented in Table 3. From these results, we observe that the proposed Dual-Encoder model, combined with the baseline method, struggles with OOD data, achieving only

a table accuracy of 0.10. This indicates overfitting to the in-domain dataset. A similar performance decrease is seen with the ColBERT configuration and the baseline, obtaining a table accuracy of 0.47. While this decrease is less significant compared to the Dual-Encoder, it still highlights the posibility of Bi-Encoders to overfit on in-domain data. The Sparse BM25+ method, however, when combined with the trained model configuration, yields the best performance on the OOD dataset, with a table accuracy of 0.60. Furthermore, the results again portray the effectiveness of the model-based approach. For instance, the table accuracy of the dual encoder improved significantly from 0.10 to 0.33 when combined with a fine-tuned Mistral model. Finally, for this dataset, instead of combining the Dual-Encoder with the untrained Mistral (Section 5.2), we combined it with ColBERT for table retrieval. ColBERT significantly outperformed the Dual-Encoder on this task utilizing the baseline query generator. The results for this combination also highlight the necessity of training the LoRA parameters, as a table accuracy improvement of 0.12 can be observed.

Table Retriever	Query Generator	TAB Acc	MSR Acc	DIM F1	RS
BM25+	Baseline	0.53	0.49	0.38	0.46
GroNLP Cosine		0.18	0.18	0.23	0.21
Dual-Encoder		0.10	0.11	0.17	0.25
ColBERT		0.47	0.44	0.34	0.48
OpenAI-ada Cosine		*0.52*	*0.45*	*0.39*	*0.47*
BM25+	Mistral - 7B	**0.60**	**0.53**	0.36	0.47
Dual-Encoder	(LoRA)	0.33	0.27	0.27	0.32
ColBERT		0.56	0.50	**0.37**	**0.51**
ColBERT	Mistral - 7B	0.44	0.35	0.29	0.43

Table 3: The results for the different Table Retrievers and Query Generators on the OOD dataset. The Mistral model in combination with the BM25+ yield the highest results.

6 Conclusions

The research presented in this paper investigates the development of an end-to-end, knowledge-graph QA system for official statistics. The system aims to tackle the challenge of retrieving a single statistical value from the vast CBS data pool based on a natural language query. The proposed approach consists of three components. First, a data augmentation method generates synthetic data based on manual pre-annotated samples. The obtained data from this step makes it possible to develop a query generator on a wider ranger of tables than relying on manually annotated data only. Secondly, a newly designed entity retrieval system has been implemented. This system retrieves the most relevant tables, measures, and dimensions in response to user queries. The results in Section 5.1 showcase the effectiveness of the proposed table retrieval models on in-domain tables, outperforming state-of-the-art embedding models from OpenAI. However, as presented in Section 5.3, these Bi-Encoder models, particularly the Dual-Encoder, struggle to generalize to unseen tables. For out-of-domain data, a traditional sparse BM25+ method is preferred. Finally, an innovative LLM-based constrained query generator is introduced. This parser utilizes retrieved entities to construct a logical expression in the form of an S-expression, which is then used to query the knowledge graph. The results in Sections 5.3 and 5.2 highlight the benefits of training LoRA parameters for this task, with significant performance improvements across all metrics compared to an untrained Mistral model and the baselines.

One overall discussion point to be made is the fact that a single query can be answered by multiple table cells, as hinted by comparing accuracy and F1 scores with the relevance scores in Section 5.2. This suggests that the other metrics might not effectively capture the capabilities of the different model configurations. Future work could explore developing a better evaluation metric for this task; one that more accurately showcases its performance in a real-life scenario. Furthermore, due to computational limitations, the Mistral-based query generator could only process up to 10 table IDs for each query. It would be interesting to see if improvements can be made when processing up to 20 tables for each query.

One limitation of the current system is that it always attempts to returns a single table cell, even for nonsensical questions or for those which CBS does not have information about. To address this, future work could explore a system for determining answerability to a given question. This would allow the model to assess its ability to answer a question as given and make suggestions to alter a question based on the available data.

Lastly, the proposed end-to-end KGQA pipeline was trained and evaluated solely on the CBS dataset presented in this paper. To effectively assess its performance, future work could explore applying the proposed pipeline to various benchmarks in this field, and/or other system implementating datacubes that can be represented using a measure-dimension like structure.

7 Impact and deployment

For reproducibility, the code and data for this paper are available online[2]. As a proof of concept, our system shows that it is possible to create a question answering system that is faithful to the CBS data and will not hallucinate, regardless of the discussed expression decoding methods used. Incorporating this system as part of a search engine can help a user get to a desired answer significantly faster.

In order to create a production-ready system that could be integrated as a search page functionality, a few steps need to be taken. First of all, used in its current form, an answer is always attempted based on a best effort, regardless of the input question. Albeit KG-faithful, it would still be undesirable to return a nonsensical answer to a question. Therefore, the system must be finetuned and incorporate a confidence threshold that can determine whether it is appropriate to return a generated answer.

Secondly, we would recommend the greedy baselines as a viable option to continue optimizing for the current form of generating single-value S-expression functions. Compared to the machine learning models, the baselines require significantly less processing power and complexity. The models can also be considered as 'black boxes', where using the baselines improve the explainability of the system. Looking forward to the upcoming registry for algorithms[3] and the act for algorithm transparency at Dutch governmental institutions, as announced by the secretary of state for digitization [15], the importance of this aspect cannot be understated. When considering more complex S-expression functions, however, the current greedy baseline would need to be altered such that multiple aggregation functions can be considered by the model.

Finally, as stated in Section 6, this system is made to be easily adoptable on other (statistical) datasets, as well as different formats like the industry-wide standard for statistical metadata, SDMX [7]. This would enable CBS as well as other institutes to make their reliable data more accessible and provide factual information to governments, citizens and the scientific community.

[2] https://github.com/Jonaskouwenhoven/Thesis-Enhancing-Graph-QA
[3] https://algoritmes.overheid.nl/en (Accessed: 26-04-2024)

References

[1] L. Bonifacio, H. Abonizio, M. Fadaee, and R. Nogueira. InPars: Unsupervised dataset generation for information retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2387–2392, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531863.

[2] W. de Vries, A. van Cranenburgh, A. Bisazza, T. Caselli, G. v. Noord, and M. Nissim. BERTje: A Dutch BERT Model. arXiv:1912.09582, Dec. 2019.

[3] Y. Gu and Y. Su. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1718–1731, Gyeongju, Republic of Korea, Oct. 2022. International Committee on Computational Linguistics.

[4] Y. Gu, S. Kase, M. Vanni, B. Sadler, P. Liang, X. Yan, and Y. Su. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*. ACM, apr 2021. doi: 10.1145/3442381.3449992.

[5] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[6] Z. Huang, S. Xu, M. Hu, X. Wang, J. Qiu, Y. Fu, Y. Zhao, Y. Peng, and C. Wang. Recent trends in deep learning based open-domain textual question answering systems. *IEEE Access*, 8:94341–94356, 2020.

[7] ISO 17369:2013. Statistical data and metadata exchange (SDMX). Standard, International Organization for Standardization, Geneva, CH, Jan 2013.

[8] ISO/IEC 20802-1:2016. Information technology — Open data protocol (OData) v4.0 — Part 1: Core. Standard, International Organization for Standardization, Geneva, CH, Dec 2016.

[9] V. Jeronymo, L. Bonifacio, H. Abonizio, M. Fadaee, R. Lotufo, J. Zavrel, and R. Nogueira. Inpars-v2: Large language models as efficient dataset generators for information retrieval. *arXiv preprint arXiv:2301.01820*, 2023.

[10] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023.

[11] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[12] O. Khattab and M. Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

[13] L. Nan, Y. Zhao, W. Zou, N. Ri, J. Tae, E. Zhang, A. Cohan, and D. Radev. Enhancing text-to-SQL capabilities of large language models: A study on prompt design strategies. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[14] OData v4.01. OData Version 4.01. Part 1: Protocol. Standard, OASIS international open standards consortium, Apr 2020.

[15] S. of state for digitization. Stand van zaken algoritmeregister (letter no. 2022-0000693912), Dec 2022. https://www.rijksoverheid.nl/documenten/kamerstukken/2022/12/21/kamerbrief-over-het-algoritmeregister Accessed: 16-07-2023.

[16] S. of state for digitization. Aanbiedingsbrief (overheidsbrede) visie op generatieve ai (letter no. 2024-0000026119), Jan 2024. https://open.overheid.nl/documenten/1be13408-2601-4dae-93c6-3c962434c616/file Accessed: 18-04-2024.

[17] OpenAI. Fine-tuning. URL https://platform.openai.com/docs/guides/fine-tuning/fine-tuning-examples.

[18] OpenAI. Introducing ChatGPT, Nov 2022. https://openai.com/blog/chatgpt Accessed: 14-06-2023.

[19] OpenAI. GPT-4 Technical Report, 2023.

[20] B. Qin, B. Hui, L. Wang, M. Yang, J. Li, B. Li, R. Geng, R. Cao, J. Sun, L. Si, F. Huang, and Y. Li. A survey on text-to-sql parsing: Concepts, methods, and future directions, 2022.

[21] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[22] P. Schneider, M. Klettner, K. Jokinen, E. Simperl, and F. Matthes. Evaluating large language models in semantic parsing for conversational question answering over knowledge graphs. *arXiv preprint arXiv:2401.01711*, 2024.

[23] Y. Shu, Z. Yu, Y. Li, B. F. Karlsson, T. Ma, Y. Qu, and C.-Y. Lin. Tiara: Multi-grained retrieval for robust question answering over large knowledge bases. In *The 2022 Conference on Empirical Methods in Natural Language Processing*. ACL, December 2022.

[24] E. Smirnov, A. Melnikov, A. Oleinik, E. Ivanova, I. Kalinovskiy, and E. Luckyanets. Hard example mining with auxiliary embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 37–46, 2018.

[25] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

[26] D. Yu, S. Zhang, P. Ng, H. Zhu, A. H. Li, J. Wang, Y. Hu, W. Wang, Z. Wang, and B. Xiang. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases, 2023.

[27] M. Zhang, O. Press, W. Merrill, A. Liu, and N. A. Smith. How Language Model Hallucinations Can Snowball, 2023.