Generative Expression Constrained Knowledge-based decoding for Open data

Lucas Lageweg[1] and Benno Kruit[2]

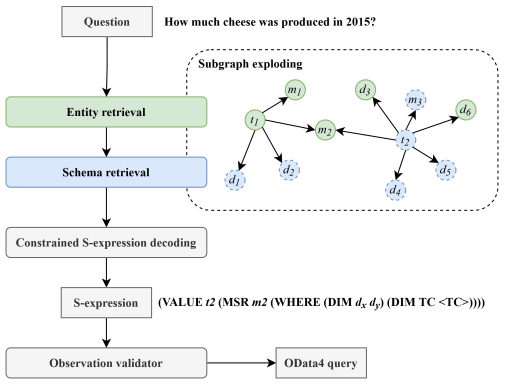
[1] Statistics Netherlands, Henri Faasdreef 312, Den Haag, Netherlands
`l.lageweg@cbs.nl`
[2] Vrije Universiteit Amsterdam, De Boelelaan 1105, Amsterdam, Netherlands
`b.b.kruit@vu.nl`

Abstract. In this paper, we present GECKO, a knowledge graph question answering (KGQA) system for data from Statistics Netherlands (Centraal Bureau voor de Statistiek). QA poses great challenges in means of generating relevant answers, as well as preventing hallucinations. This is a phenomenon found in language models and creates issues when attempting factual QA with these models alone. To overcome these limitations, the Statistics Netherlands' publicly available OData4 data was used to create a knowledge graph, in which the answer generation decoding process is grounded, ensuring faithful answers. When processing a question, GECKO performs entity and schema retrieval, does schema-constrained expression decoding, makes assumptions where needed and executes the generated expression as an OData4 query to retrieve information. A novel method was implemented to perform the constrained knowledge-based expression decoding using an encoder-decoder model. Both a sparse and dense entity retrieval method were evaluated. While the encoder-decoder model did not achieve production-ready performance, experiments show promising results for a rule-based baseline using a sparse entity retriever. Additionally, the results of qualitative user testing were positive. We therefore formulate recommendations for deployment help guide users of Statistics Netherlands data to their answers more quickly.

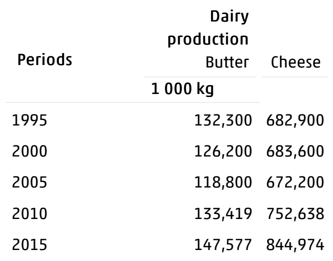
1 Introduction

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, we have observed that non-experts currently struggle to find the correct tables for their needs in the vast amount of data available. The current 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 [1] and GPT4 [2] getting much attention. Due to the nature of these models, they are able to generalize very


Question
How much cheese was produced in 2015?
Subgraph exploding
Entity retrieval
Schema retrieval
Constrained S-expression decoding
S-expression
(VALUE t2 (MSR m2 (WHERE (DIM d_x d_y) (DIM TC <TC>))))
Observation validator
OData4 query
m1
m2
m3
t1
t2
d1
d2
d3
d4
d5
d6

(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

Fig. 1: (a) Overview of the GECKO pipeline from query to answer. Candidate nodes (green) are retrieved for the query and used for doing subgraph exploding (blue), after which it 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.

well on a large range of topics, but have shown to be prone to 'hallucinating', where plausible but incorrect or even nonsensical answers are being generated [3]. Especially for official data like governmental statistics, this is highly undesirable behaviour. Therefore, the main design goal for this system is the interpretability of the provenance of answers.

Knowledge graph question answering (KGQA) is a field where knowledge bases (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 **GECKO** (**G**enerative **E**xpression **C**onstrained **K**nowledge-based decoding for **O**pen data), a proof of concept for a generation-based KGQA system for CBS data [3]. It will generate a response to a question in two parts: the answer itself following from the KG and a justification that should be able to explain why and what exactly is returned. The justification is crucial, as the KGQA system could return facts that, albeit correct, are not relevant for the question.

We focus on the retrieval of a single table cell from a relevant table based on a given natural language input question. In practise, this means that the system will be able to answer questions that have an answer in a single cell of a table present in the KG. For our investigation, we assume only Dutch input questions, and focus on all of the Dutch '*key figure*' tables in the CBS data catalogue. These

[3] The source code, graph and models are made available at `https://github.com/lagewel001/GECKO`.

60 tables, ranging over all different topics, contain the highest aggregated form of statistical data available at the CBS. This makes these tables suited for our specific proof of concept as this high-level data has little ambiguity.

As this project is still in the early stages of preparation for deployment, we present preliminary results of the application of these semantic technologies to this problem, and their fitness for purpose. In short, our contributions are as follows:

- we describe our Knowledge Graph Question Answering application on statistical observation data from Statistics Netherlands (Section 3),
- we evaluate the application (Section 4), both quantitatively on in-house annotated data (Section 4.1) and qualitatively with user tests (Section 4.2),
- and we discuss the impact and deployment of the application (Section 5).

2 Related Work

Systems mostly related to our approach are semantic parsing based QA systems, in particular text-to-SQL. In their survey, Qin et al. [4] explain several approaches on learning input and table schema representations (encoding), in order to later generate and parse SQL statements (decoding). Also, in earlier attempts at CBS, research was done on creating a text-to-SQL model for retrieval of micro-data (non-aggregated) statistics [5]. While similar to our approach in their query expression design, our system is specifically designed to tackle the case where a large number of heterogeneous tables and metadata concepts must be retrieved and used in queries. This quality is shared by KGQA systems, which aim to retrieve one or more small-scale facts (i.e. RDF triples) from expansive KBs, often using analogous text-to-SPARQL approaches. Damljanovic et al. [6] and Unger et al. [7] construct SPARQL templates and parse the questions using those templates. The downside of using templates is its limited flexibility, as only question types for which templates are created can be answered and thus no true free-form QA can be achieved. However, free-form generation of SPARQL (e.g. Ochieng [8]) is hard, as the syntax is quite abstract and intricate in the representation of data triples, and queries can become quite expansive. As the surveys by Lan et al. [9] and Gu et al. [10] show, more recent approaches use either ranking-based methods or generation-based approaches for creating alternative logical forms for querying the KG. Similarly, we generate simplified logical S-expressions, which are translated later into more complicated query language constructs.

Large Language Models (LLMs) have recently gained a lot of attention as free-form text generators that can be used as QA systems [1]. One major drawback of these systems is their tendency to generate false statements. Fact checking using KBs has been proposed as one solution [11]. Generative models are also considered for the task of generating logical forms to retrieve and reason over KBs 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. [12], Yu et al. [13] and Shu et al. [14]. All three examples take roughly the same overall approach, and serve as the main source of inspiration for this current research.

3 Application

Data For this research, to narrow the scope, we will use CBS data that is publicly available and, more specifically, will use the *key figure* tables, which contain the most aggregated form of statistics. A complete overview of all different statistical research done by CBS is published yearly [15]. All data is made available via the ISO/IEC approved Open Data Version 4.01 protocol (OData4) [16,17], 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, in which every unique measure and dimension has an identifier[4]. 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. However, this makes it possible for a single identifier to have multiple nuanced definitions based on the tables it is part of. For example, the standardised terms *Total imports of goods and services* and *Perception of (un)safety* have multiple definitions depending on how the goods/services are exactly defined and what specific question is asked to an respondent, which can vary in different surveys. This can also happen when a new but very near identical definition of a term is inadvertently added in the redaction process. In this work, we make use of a subset of this vocabulary encoded as RDF as our KG, corresponding to the schema descriptions of the tables in our target sample.

Knowledge Graph Using RDF to represent the table structures creates a simple and intuitive system to work with for referencing observations later. Due to the tabular nature of the data at hand, and by only indirectly storing the references to observations using the tables' metadata, the KG remains relatively simple. Currently CBS makes its data available using the OData4 format, while the Statistical Data and Metadata eXchange (SDMX) format is currently the industry standard [18]. When transitioning to SDMX in the future, or when using SDMX for a different use case, our vocabulary can be used for setting up a KG that can be used by the subsequent methods that will be discussed in the following sections. This implies that our method can be applied by anyone using one of these standards.

During the creation of the KG, whenever applicable we attempted to harmonize as many measures and dimensions as possible. This improves the system's metadata overall but will also help simplify the generation process for querying the KG as this results in less nodes overall that can cause ambiguity when matching to a query. This is an iterative process as new tables, measures and

[4] `https://vocabs.cbs.nl/en/`

dimensions are constantly being created and new instances are found where a harmonization of two nodes can be beneficial. For example, at the time of writing there are two separate nodes/identifiers for the term *labour costs*, with one used for the labour cost survey and the other for production statistics. As said, nodes can contain more than one definition based on the nuances of the table and as this research puts its focus on the key figure statistics, undoubtedly a lot of harmonization can be done over the entirety of the CBS' data catalogue. Furthermore, this harmonization will also help in the editorial process as this enables the statisticians and editors to select standardised predefined options and only need to create new measures or dimension when no standardized code is available. One example of this in practise can be found in the standardization of CBS municipality codes (among other geographical related entities), which are also publicly shared in Wikidata[5], to be used by third parties to refer to related CBS data. Other examples include the use of the European-wide Classification of Products by Activity (CPA) [19] and Statistical classification of economic activities (NACE) [20].

Queries In order to retrieve and return information from the KG, several solutions can be considered. For example, a single prefix tree (trie) can be returned as plain text, or a part of the KG can be returned in the form of the aforementioned RDF-format or bindings of a SPARQL query. Several other solutions to provide semantically meaningful representations for KGQA have been proposed, like graph query [21] or λ-calculus [22]. In this work we follow ArcaneQA's solution based on GrailQA's S-expression format [12] to provide a syntactic sugar for queries that can be executed over the KG.

S-expressions are a notation for logical expressions containing atoms and expressions (which are always S-expressions themselves) in a tree-like structure as nested lists. For our purpose, GrailQA proposes 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 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. As our use case is quite different from ArcaneQA's, we use a modified version of S-expressions more suited to our KG. The S-expressions should be able to represent one or more OData4 observations. In practice, this means that the expression must be able to notate at least a table, measure and one or more dimensions to represent a single observation. Appendix A gives a supplementary table with the set of functions created for this purpose. 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. Here, only a `VALUE` function is needed to retrieve a raw observation, but more functions for

[5] `https://www.wikidata.org/wiki/Property:P382` (Accessed: 14-07-2023)

counting, averaging, etc. can be considered later. Two special atom placeholders exist for the time and geographical constraint functions (`TC` and `GC`).

Pipeline The GECKO pipeline consists of four parts, described in Fig. 1. This pipeline is mostly inspired by the similar format shown by Shu et al. [14]. Due to its large size, it is infeasible to consider the entire graph when doing KGQA. We restrict the querying space by performing entity retrieval based on the query to determine relevant graph nodes. Two methods of entity retrieval were implemented: (1) Sparse retrieval, using BM25+ [23] to rank candidate nodes as a baseline. It takes the top 25 best ranking table, measure or dimension nodes based on their textual metadata (excluding time or geographical dimensions) and uses them for schema retrieval (see next step). The documents here are the table, measure and dimension entities and use the concatenated metadata of `skos:prefLabel`, `skos:altLabel`, `skos:definition` and `dct:description` as the search body. Dimension entities of type `TimeDimension` and `GeoDimension` are not considered in the search to significantly reduce the search space in the following steps. This method of sparse retrieval has shown to still be very competitive compared to more complex embedding-based dense search methods [24]. (2) Dense vector search, using context embedding vectors of the KG nodes in an inner product (IP) index and embed the given query to compare in the vector space to return the 75 closest nodes. The context embeddings are created using a sentence transformer [25] pre-trained language model (PLM) on the same textual metadata fields as mentioned above.

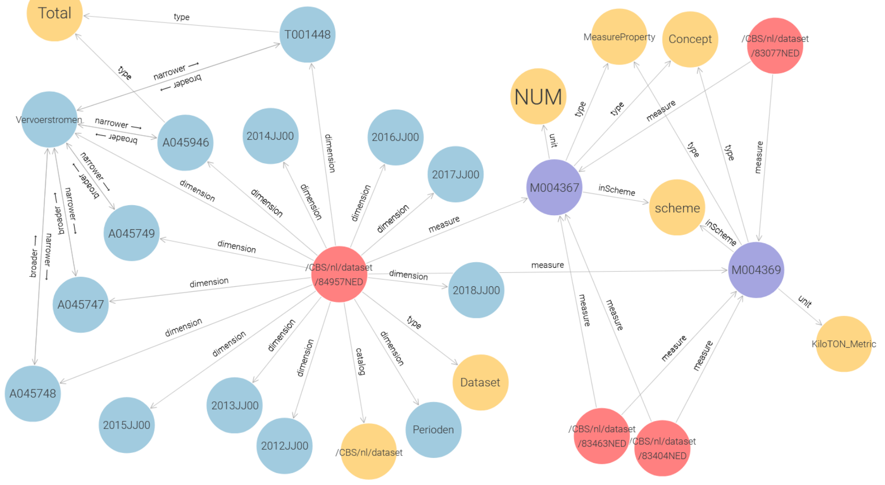


Fig. 2: Exploded subgraph for a small CBS table (`84957NED`: Pijpleidingenvervoer; kerncijfers) as visualized in GraphDB, containing the table nodes (red), dimensions and their hierarchical relations (blue), measures (purple) and several properties (yellow) like units.

After obtaining the closest matching entities based on the query, we use schema retrieval to explode a subgraph using the entities given (Fig. 2). The result of the subgraph exploding is a graph containing all table nodes and their related measures and dimensions having nodes intersecting with the retrieved entities. This is visualized in Fig. 1a. The green nodes are those obtained using the entity retrieval step. The resulting graph from the schema retrieval are all depicted nodes, where the blue nodes are obtained trough the subgraph exploding. This results in a subgraph containing all relevant nodes connected to one or more tables. Relevant (hierarchical) relations and useful metadata for the different entities are also retrieved. Both the question and subgraph are input for the constrained S-expression generation. This step results in an expression as described above.

Observation validation The observation validator translates the generated S-expressions to an OData4 query. As a single OData4 observation requires the query to explicitly state all dimension group filters, dimensions not included in the expression are assumed if possible using the table subgraph. For missing dimensions of a time or geographical type, the latest period available up until the current year or the biggest aggregate of the measure available (usually the Netherlands) are assumed respectively. In any other case, the validator checks if the group contains a dimension denoting a total. If no assumption can be made, the user should be asked to refine the given query and specify the missing dimension group. All assumptions made are included in the justification output to the user. S-expressions can contain a `<TC>` or `<GC>` placeholder. As there is rarely ambiguity between dates or geographical locations, these can be substituted by rules with the relevant dimension entities based on the subgraph and the given query. As ambiguity is still possible however, the largest dimension aggregation is chosen similar to the default assumptions if not specified in the question. This system also enables relative notations like 'last year' to be substituted correctly based on the current date. The final step in the pipeline is the execution of the OData4 query by translating the final expression to an OData4 query. The justification consists of the question, table, measures and dimensions selected and the default assumptions made.

Example Following the example from figure 1, the following expression can be generated from the question "how much cheese was produced in 2015?":

```
   (VALUE (7425zuiv (MSR D001544 (WHERE (DIM TC <TC>)))))
```

Which will be rule-based substituted by the obervation validator into:

```
   (VALUE (7425zuiv (MSR D001544 (WHERE (DIM TC 2015JJ00)))))
```

And finally translated to its OData4 query counterpart:

```
   https://odata4.cbs.nl/CBS/7425zuiv/Observations/?$filter=Measure in ('D001544')
and Perioden in ('2015JJ00')
```

PLMs As one of the preliminaries for creating our model we employ pretrained language models (PLMs) in two instances. The first is in creating the context embedding vectors for the dense vector search in entity retrieval as described

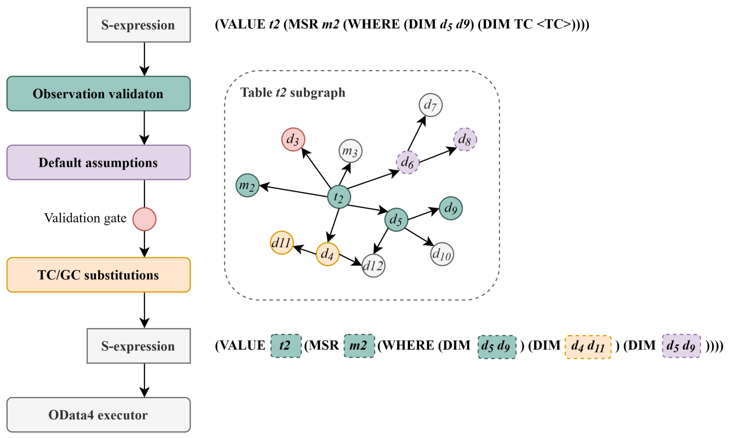


Fig. 3: Observation validation and substitutions of missing measures and dimensions in the generated S-expression. From initial nodes (teal), missing dimensions for observations (purple) are assumed if possible, and time and geographical constraints are filled in using rules (orange). If a required dimension is missing (red), validation fails.

above. These embeddings are created using a pretrained Dutch BERTje-based sentence transformer [26]. We will refer to this PLM as the GroNLP model. The second case is as a starting point for finetuning the constrained decoder model. For our model training, we finetuned the Dutch RoBERTa model, RobBERT [27], on a cleaned CBS dataset containing all written publications, articles and table descriptions. This was done using Masked Language Modeling (MLM). We will refer to our finetuned model as SNERT.

3.1 Generative models

A rule-based baseline was created for generating S-expressions to compare our model against. The S-expressions can be created token-by-token such that, given the subgraph, admissible tokens can be returned at every step. The baseline uses the BM25+ scores from the entity retrieval step to greedily determine what token from the admissible tokens to select. This deterministic approach implies that only the best scoring table and its measures and dimensions can be selected for the expression generation. Only scored nodes are considered for the expression output. Therefore, after the schema retrieval step, the BM25+ scores are calculated for all nodes in the subgraph to increase the recall. Scores of measures and dimensions denoting a total are boosted. All dimensions in the set of admissible tokens are greedily added to the expression. Therefore, a minimum threshold score per node is implemented to prevent dimensions with low scores to be considered in the output.

For the model solution, generation of the expressions is performed by a transformer-based seq2seq model. Utilising the encoder-decoder framework, the model will output the target S-expression by generating sequences of admissible tokens until a valid expression is constructed. We employ the core idea of dynamic contextualised encodings from Gu et al. [28] for encoding representative contextual embeddings for the code tokens. First, we extend the model's tokenizer vocabulary with all code nodes present in the KG. We will denote this extension to the SNERT tokenizer as SNERTe. Consequently, when initialising the model, the embedding weight matrix for these tokens are initialised randomly. Secondly, an index map is created, mapping all token identifiers to their respective position in the pre-computed IP-index, which contains all embeddings for the different code nodes based on their textual descriptions. Next, we update the embedding weight matrix for all code nodes to utilise the weights from the IP-index. In all following steps and during training, we fix the gradients for the embedding layer. This way, the decoder can learn the relations between the input question and given vectors from the schema retrieval step. The individual context vectors of nodes thus remain constant.

As the embedding layer, as part of the encoder, is not tasked with learning and creating the different embeddings for the KG nodes, new or altered codes only need to be added to the IP-index and the vocabulary. Our hypothesis is that this will help the model generalize, and:

($\mathbf{H_1}$) reduce the amount of training data needed for successfully training the model to recognize relations between questions and entities with the limited number of training samples available (learnability);

($\mathbf{H_2}$) solve the infeasibility of learning all input-output relations between questions and all possible codes in the KG (generalizability);

($\mathbf{H_3}$) and solve the infeasibility to retrain the model every time new nodes are added to the KG (maintenance).

For an incoming question a dynamic prompt is generated. We employ entity and schema retrieval as before using the dense vector search. For performance benefits during the subgraph exploding and to prevent overflowing the input prompt, we consider only the top 5 candidate table nodes, and for each table the top 10 scored measures and dimensions to add to our prompt. The question and KG tokens are then concatenated into the following sequence:

$$\mathrm{[CLS]}, \mathrm{q}_1, ..., \mathrm{q}_{|\mathrm{k}|}, \mathrm{[SEP]}, \mathrm{t}_1, |\mathrm{MSR}|, \mathrm{m}_1, ..., \mathrm{m}_{|\mathrm{p}|}, |\mathrm{DIM}|, \mathrm{d}_1, ..., \mathrm{d}_{|\mathrm{q}|}, \mathrm{[SEP]}, \mathrm{t}_{|\mathrm{n}|}, ...$$

where $\{\mathrm{q_i}\} \subset \mathrm{Q_t}$ denotes all wordpiece tokens following from the tokenizer and $(\{\mathrm{t_i}\} \cup \{\mathrm{m_i}\} \cup \{\mathrm{d_i}\}) \subset \mathrm{W_t}$ denotes respectively all table, measure and dimension tokens ordered by their graph relations and vector distances compared to the embedded query in the IP-index. Dimensions regarding time and geographical constraints are omitted from the prompt and are substituted with `<TC>` and `<GC>` in the target expressions for the model. In fact, these tokens are not added to the tokenizer's vocabulary and model's embedding matrix to avoid the model becoming too large, as there is an incredibly large number of these nodes in the KG.

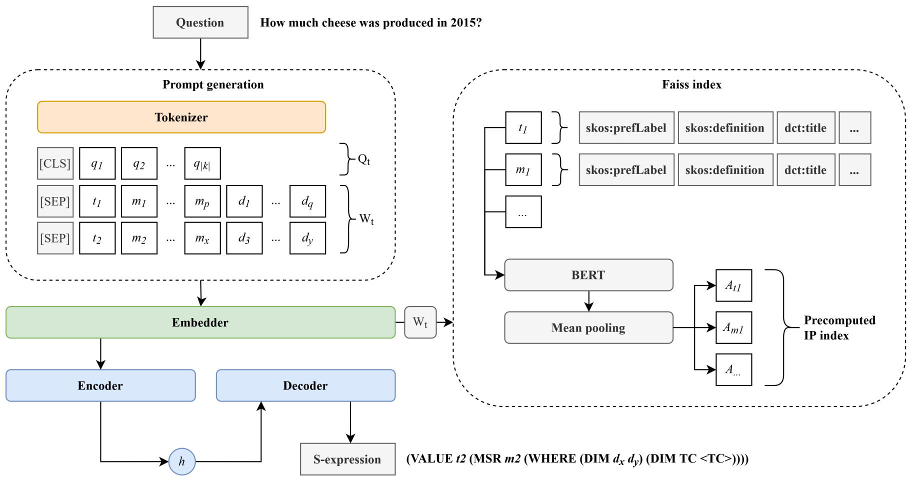


Fig. 4: Overview of the GECKO constrained S-expression encoder-decoder model. The question is tokenized into question tokens Q_t and candidate nodes W_t. At the embedding step, for all tokens in W_t the contextualised embedding matrix is obtained from the pre-computed Faiss index.

Using the different PLMs, the model is trained with the embedded prompt as input, and the S-expression as target sequence, using weighted cross-entropy with label smoothing as the objective function. As the opening and closing parentheses are the most predominant and constant tokens in the target expressions, the loss function is weighted for these specific tokens. Considering an extended vocabulary K, we define a vector ω and a weight parameter β to be

$$\omega_k = \begin{cases} \beta, & \text{if } k \in \{\texttt{[CLS]}, \texttt{[SEP]}, \texttt{[PAD]}, \texttt{(}, \texttt{)}\} \\ 1.0, & \text{otherwise} \end{cases}, \quad \text{with} \quad \beta \in [0, 1].$$

We also attempt to reduce model overconfidence in the beam search inference by applying label smoothing regularization [29]. Combined, this yields the following objective loss function:

$$\mathcal{L} = -\sum_{i=1}^{n} \sum_{k \in K} \omega_k q'(k|y_i) \log p(k|y_i),$$
$$\text{with} \quad q'(k|y_i) = (1-\epsilon) q(k|y_i) + \frac{\epsilon}{K}$$
$$\text{and} \quad p(k|y_i) = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}$$

where y denotes the generated target expression, n the length of this sequence, q the ground truth (i.e. one-hot vector for every y_i) and p the softmax function for a given target token, denoting z as the logit for a given token. ϵ denotes the smoothing parameter in the regularization function q'.

Constrained inference As the decoder is not strictly constrained during the training phase, in contrast to ArcaneQA's approach, we perform constrained decoding during inference to prevent generating faulty sequences. This ensures that the generated S-expressions are syntactically correct and true to the KG. We employ the beam search algorithm [30] to generate the top-ranking sequences, i.e. expressions, over n beams. Using the constrained beam search method from De Cao et al. [31] we force the decoder to generate only admissible tokens following from the current S-expression and the exploded subgraph. In practice, this means that when a specific table identifier is generated for a specific beam, only measures and dimensions that have a relation with that table node can be passed as admissible tokens at a specific timestep t.

4 Evaluation

For determining the model performance we test the hypotheses H_1, H_2 and H_3 stated above. The evaluation dataset consists of 120 unseen samples from our annotated key figures dataset. For the generalization/scalability test we let several annotators familiar with the data create ±1.250 questions for the top-visited non key figure tables. These tables are unknown to the model and contain measures and dimensions not trained on. For the annotation task, we provided an interface serving a pivot table containing a single randomized OData4 observation and asked the annotators to come up with one or more corresponding questions.

To evaluate the entity retrieval step we will use the accuracy (Acc.), precision (P), recall (R) and F1-scores of the tables, measures and dimensions. These are calculated based on the generated prompts for a given query compared to the corresponding S-expression, using BM25+ and dense vector search. Mainly, a high accuracy or recall is desired for a correct outcome of the expressions in the generation step, as the model should learn to select the correct codes from a prompt into the generated expression. The same applies for the BM25+ baseline, as the code selection is performed using a greedy approach. For table and measure nodes, the accuracy scores denote the proportion of target nodes occurring in the input prompts, regardless of the length of the prompt. The precision scores also take into account how many table, measure or dimension nodes there are in the prompt (i.e. codes that are in the prompt but might not occur in the target expression). For tables, the mean reciprocal rank (MRR) is also calculated to indicate to what degree the correct tables and their selected measure and dimension codes occur at the start of the prompt, as the highest scoring tables from the entity retrieval step are placed at the beginning of a prompt.

To determine the performance of the baseline and trained model we use the ROUGE [32] and BLEU [33,34] metrics, often used in machine translation. This will determine the expressions' recall and precision respectively. To also syntactically evaluate matching expressions (i.e. correct relative placement of parentheses and functions), but disregard dimension filter ordering, the bi-gram version of ROUGE will be used alongside four-gram BLEU. For the individual table and measure codes in a generated S-expression, the exact match (EM) score will be used, as there can be only one of each in a given S-expression using the current expression format. The EM score therefore denotes the proportion of identical table or measure nodes between all target-prediction expression pairs. For the dimension tokens, of which there can be multiple in a single expression, we determine the F1 score.

Additionally, we use a human-judgement relevancy score (RS), manually annotating every model answer retrieved from OData4 following the generated expressions with either 1 (fully correct), 0.5 (minor issues) or 0 (wrong). All models were trained on 947 training samples from the key figures dataset, using 120 samples for evaluation. For our last configuration, we trained the SNERTe PLM on all samples available (key figures plus annotated top-visited non key figures), counting 2.069 training and 230 evaluation samples.

An important part of the evaluation is to test the generalizability of the model. As the KG and number of unique codes is vast, it is undesirable and unfeasible to annotate samples for all possibilities by hand. Next to that, frequent updates, additions and changes occur to the KG which can make the trained model outdated. To test our generalizability and maintenance hypotheses (see H_2 and H_3) we evaluate the model's generalizability and scalability on our metrics using controlled samples of S-expressions referencing never before seen tables. These tables and corresponding nodes were omitted completely from the KG during training and are added to the vocabulary before inference using the method described above.

4.1 Quantitative Results

Table 2 shows the evaluation scores of the table, measure (MSR) and dimension (DIM) nodes from the entity retrieval and prompt generation steps. Both the sparse (BM25+) and dense retrieval methods perform similarly, showing slight deviations in matching accuracy and precision per node type. Neither method significantly outperforms the other based on these scoring metrics. The overall results of the model performances can be seen in Table 1. Of the model-based solutions, the RobBERT-based model scores best on our evaluation set on both the ROUGE-2, BLEU, and matching criteria save from MSR EM, which is scored best by the SNERT-based model. The SNERT-based model trained on all data samples scores similarly to the smaller model during training.

The evaluation of the baseline model shows the highest scores on all metrics, significantly outperforming the model-based approaches. Both the sparse and dense vector search methods were tested for the entity retrieval (ER) step, with BM25+ scoring slightly higher on all measures except for table matching. Regarding the RS-score, for 80 samples, the BM25+ baseline test resulted in 27 relevant (=1) and 3 semi-relevant (=0.5) answers. The dense retrieval scored 21 and 11 in these categories respectively. This difference is also reflected in the similar RS and EM scores, but significantly different dimension F1.

Model	ER	ROUGE-2	BLEU	RS	Table EM	MSR EM	DIM F1
Baseline	BM25+	**0.437**	**62.198**	**0.378**	0.347	**0.198**	**0.621**
	Dense	0.374	53.025	0.349	**0.396**	0.158	0.496
GroNLP	Dense	0.294	48.039	0.107	0.181	0.029	0.455
RobBERT	Dense	0.377	55.042	0.110	0.267	0.038	0.555
SNERTe	Dense	0.193	40.278	0.031	0.200	0.048	0.214
SNERTe (all samples)	Dense	0.318	46.182	0.167	0.188	0.100	0.398

Table 1: Evaluation metrics and S-expression inference evaluation for the different models. The target-prediction expression similarity is expressed by ROUGE-2, BLEU, F1 and exact match (EM) scores. The relevancy score (RS) is the average of manually annotated relevant answers following from the questions and generated expressions by the model (i.e. disregarding exact target matches).

		BM25+	Dense
TABLE	**Acc.**	**0.530**	0.496
	P	0.114	**0.184**
	MRR	**0.262**	0.239
MSR	**Acc.**	**0.448**	0.435
	P	0.018	**0.074**
DIM	**P**	0.023	**0.074**
	R	**0.592**	0.555
	F1	0.040	**0.054**

Table 2: Evaluation results for entity retrieval performance using individual metrics for table, measure and dimension nodes by BM25+ and dense vector search.

Model	ER	RS	Table EM	MSR EM	DIM F1
Baseline	BM25+	**0.357**	**0.409**	**0.278**	**0.564**
	Dense	0.182	0.178	0.105	0.358
GroNLP	Dense	0.081	0.126	0.039	0.176
RobBERT	Dense	0.066	0.114	0.027	0.223
SNERTe	Dense	0.055	0.076	0.013	0.170

Table 3: Inference evaluation for non-key figure (unseen) tables for the different models. Target-prediction similarity is expressed by the exact matches (EM) of tables and measures nodes and the F1 score for dimensions. RS denotes the relevancy scores of answers generated. All models are significantly outperformed by the rule-based baseline.

Noteworthy is the low precision and relatively high recall on the entity retrieval for dimensions, indicating the input prompts span a high number of dimensions (lowering the precision), containing for more than half of the samples the correct target dimension(s). This can also be seen in the resulting F1 scores for the dimensions in the S-expression evaluation, where for the baseline, GroNLP and RobBERT models the F1 scores are close to 0.5.

4.2 Qualitative user testing

The test results from the previous section only show the performance on exact matches between the target and generated expressions, and the relevance of answers generated were determined based on an annotation guideline. In doing this, there remains a reality-gap between the model scores of the system and the actual usefulness to a user. To test this, there are a few questions that need to be answered. First of all, how well does a user parse the information that is presented to them when a question is answered? Is the pivot table shown as an answer an intuitive and readable way of presenting the statistical data? Secondly, how well do the users understand the justification given with each answer, and do they convey the right information when an assumption or ambiguity is present? This ties in with the question whether this approach helps to steer the users' querying of the system when unrelated or no answers are returned. Finally, as the main question, we would want to know if GECKO helps the users to find their statistical answers more quickly than without using our system.

Our user study comprised of 7 unique users. Example scenarios were constructed asking the users to find information on specific cases (e.g. finding different statistics and trends on solar power and number of solar panels in the Netherlands), as well as asking the users to bring scenarios of their own. The users have full control over what search queries to use and no external help influencing the results is provided. The overall consensus was that a system like GECKO is a very suitable option for finding statistics more quickly, especially for non-frequent users of our data. The users did indicate that the biggest drawback at the moment is the result comprising of only a single table cell, which does not help in showing what more information can be found in the table shown. This makes it impossible to do any associative searching by the user. Furthermore, it was shown that the way the answer justification is presented is vital to our use case. Presenting a textual prompt only containing the assumptions made and word matches between the query and measures/dimensions did not convince the users and raised more questions than it answered. Lastly, users would like an option to alter the assumptions made, in case a wrong default value is given, alongside the possibility to give feedback to the system in case a completely irrelevant answer is given.

5 Impact and deployment

As a proof of concept, GECKO 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, using GECKO in its current form, an answer is always attempted based on a best effort (i.e. closest match), 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 baseline as a viable option to continue optimizing for the current available S-expression functions. Compared to machine learning models, which can be considered as 'black boxes', this also improves the explainability of the system. Looking forward to the upcoming registry for algorithms[6] and the act for algorithm transparency at Dutch governmental institutions, as announced by the secretary of state for digitization [35], the importance of this aspect cannot be understated. A first step in this would be in investigating the possibilities of a reranking algorithm and combining the sparse and dense entity retrieval methods. When considering more complex S-expression functions, the current greedy baseline would need to be altered such that multiple aggregation functions can be considered by the model. As of the current state, the model-based approach does not yield reliable results for a production environment. To determine its viability, significantly more training data is needed. Synthetic training data can be considered alongside the annotated samples. We recommend conducting a cost-benefit analysis for this scenario and investigate if there is a functional requirement for being able to return more complex answers.

6 Conclusions and Future Work

In this paper we present GECKO as a question answering system to help guide users of CBS data to relevant answers for their questions. The system uses a knowledge graph containing table metadata and generates expressions that can be used for querying and retrieving observations from OData4. The results show that there is not a significant difference between the performance of the sparse and dense search methods for the entity retrieval step when it comes to exact matches of table, measure and dimension nodes compared to the questions' target expressions. When expanding the KG with more nodes however, the BM25+ sparse retrieval method outperforms the dense approach. When looking at the decoding performances of the different models compared to the baseline, the learnability hypothesis H_1 is disproven, as the models did not yield competitive results by training on the limited number of training samples available. We cannot conclude our generalizability hypothesis H_2. None of the models were able to generate more relevant answers from the candidate nodes in the prompt, and thus we cannot surely state that by fixing the embedding matrix the need of learning all input-output relations is omitted. This might be due to the limited number of training samples that were available and should thus be revisited in the future. Looking at the maintenance hypothesis H_3, we see that there is a slight drop in performance for the three different models when looking at the non-key figure results for unseen tables. As the general performances for both evaluation sets are too low however, this can neither support nor reject the claim that using the fixed embedding layer helps with generalizing over all possible

[6] `https://algoritmes.overheid.nl/en` (Accessed: 06-12-2023)

(even unseen) nodes and remains to be tested when further improvements are made.

Currently, plans are being made to expand the system to be able to work on different datasets. In its current form, a KG can be generated from any OData4-based system. In the near future, we will expand the proof of concept to accommodate SDMX-based datasets, with SDMX being the industry standard that will also be utilised at CBS in the near future.

Future work can look at the possibility for more complex S-expressions in order to allow more complex and diverse questions to be answered. Next to more complex answers, future research could also look into the possibility for determining unanswerability of questions, as the current system will always attempt a best effort to answer the question given using the closest matching KG entities. Finally, combining the benefits of both sparse and dense entity retrieval methods might increase the relevance significantly for generating S-expressions. A combined distance metric for the entity retriever or a reranking solution can both be investigated.

References

1. OpenAI. Introducing ChatGPT, Nov 2022. `https://openai.com/blog/chatgpt` Accessed: 14-06-2023.
2. OpenAI. GPT-4 Technical Report, 2023.
3. Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. How Language Model Hallucinations Can Snowball, 2023.
4. Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, Fei Huang, and Yongbin Li. A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions, 2022.
5. Tjalling Gelsema and Guido van den Heuvel. Towards demand-driven on-the-fly statistics. Preprint on webpage at `https://www.researchgate.net/publication/349768489_Towards_demand-driven_on-the-fly_statistics`, 2021.
6. Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-Based Lookup through the User Interaction. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *The Semantic Web: Research and Applications*, pages 106–120, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
7. Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-Based Question Answering over RDF Data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, page 639–648, New York, NY, USA, 2012. Association for Computing Machinery.
8. Peter Ochieng. PAROT: Translating Natural Language to SPARQL. *Expert Syst. Appl.*, 176(C), aug 2021.
9. Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. A Survey on Complex Knowledge Base Question Answering: Methods, Challenges and Solutions, 2021.
10. Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su. Knowledge Base Question Answering: A Semantic Parsing Perspective, 2022.

11. Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Lidong Bing, Shafiq Joty, and Soujanya Poria. Chain of Knowledge: A Framework for Grounding Large Language Models with Structured Knowledge Bases, 2023.
12. Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu 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.
13. Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. DecAF: Joint Decoding of Answers and Logical Forms for Question Answering over Knowledge Bases, 2023.
14. Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje F. Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew 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.
15. Centraal Bureau voor de Statistiek. Wij maken en verspreiden statistieken - Overzicht statistieken, 2022. `https://www.cbs.nl/nl-nl/over-ons/wij-maken-en-verspreiden-statistieken` Accessed: 13-07-2023.
16. OData Version 4.01. Part 1: Protocol. Standard, OASIS international open standards consortium, Apr 2020.
17. Information technology — Open data protocol (OData) v4.0 — Part 1: Core (ISO/IEC 20802-1:2016). Standard, International Organization for Standardization, Geneva, CH, Dec 2016.
18. Statistical data and metadata exchange (SDMX). Standard, International Organization for Standardization, Geneva, CH, Jan 2013.
19. Commission Regulation (EU) No 1209/2014 of 29 October 2014 amending Regulation (EC) No 451/2008 of the European Parliament and of the Council establishing a new statistical classification of products by activity (CPA) and repealing Council Regulation (EEC) No 3696/93, Oct 2014. `http://data.europa.eu/eli/reg/2014/1209/oj`.
20. Regulation (EC) No 1893/2006 of the European Parliament and of the Council of 20 December 2006 establishing the statistical classification of economic activities NACE Revision 2 and amending Council Regulation (EEC) No 3037/90 as well as certain EC Regulations on specific statistical domains, Dec 2006. `http://data.europa.eu/eli/reg/2006/1893/oj`.
21. Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 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, July 2015. Association for Computational Linguistics.
22. Qingqing Cai and Alexander Yates. Semantic Parsing Freebase: Towards Open-domain Semantic Parsing. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 328–338, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics.
23. Yuanhua Lv and ChengXiang Zhai. Lower-Bounding Term Frequency Normalization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, page 7–16, New York, NY, USA, 2011. Association for Computing Machinery.

24. Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. Simple Entity-Centric Questions Challenge Dense Retrievers, 2022.
25. Nils Reimers and Iryna 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.
26. Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. BERTje: A Dutch BERT Model. arXiv:1912.09582, December 2019.
27. Pieter Delobelle, Thomas Winters, and Bettina Berendt. RobBERT: a Dutch RoBERTa-based Language Model, 2020.
28. Yu Gu and Yu 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, October 2022. International Committee on Computational Linguistics.
29. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision, 2015.
30. Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks, 2014.
31. Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive Entity Retrieval, 2021.
32. Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.
33. Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.
34. Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics.
35. Secretary 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.

Appendices

A S-expression functions considered in GECKO

Function	**Arguments**	**Description**
`VALUE`	table entity	Returns all the raw OData4 observations (i.e. cell values) matching the table entity and corresponding selection following the expression.
`MSR`	(measure entity, `WHERE`-expression)	Denotes a selection filter for a table measure to retrieve.
`WHERE`	set of `DIM`-expressions	Function containing all sub-expressions for filtering dimensions on a specific measure.
`DIM`	(dimension group entity or `TC`/`GC` atom, dimension entity)	Denotes a selection filter for the table dimensions to retrieve.
`TC` / `GC`	dimension entity	Special temporal or geographical constraint function denoting a dimension filter for specific dimensions of type `TimeDimension` and `GeoDimension` respectively.
`OR`	set of dimension entities	Function for defining a selection filter on multiple dimension entities in the same dimension group.

Table 4: Set of different S-expression functions for our system.