Edinburgh Research Explorer

DISCOSQA: A Knowledge Base Question Answering System for Space Debris based on Program Induction

Link:
[Link to publication record in Edinburgh Research Explorer](#)

Document Version:
Publisher's PDF, also known as Version of record

Published In:
Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics

Knowledge Base Question Answering for Space Debris Queries

Paul Darm[1]
paul.darm@strath.ac.uk

Antonio Valerio Miceli-Barone[2]
amiceli@ec.ac.uk

Shay B. Cohen[2]
scohen@inf.ed.ac.uk

Annalisa Riccardi[1]
annalisa.riccardi@strath.ac.uk

[1] Mechanical and Aerospace Engineering, University of Strathclyde
[2] School of Informatics, University of Edinburgh University

Abstract

Space agencies execute complex satellite operations that need to be supported by the technical knowledge contained in their extensive information systems. Knowledge bases (KB) are an effective way of storing and accessing such information at scale. In this work we present a system, developed for the European Space Agency (ESA), that can answer complex natural language queries, to support engineers in accessing the information contained in a KB that models the orbital space debris environment. Our system is based on a pipeline which first generates a sequence of basic database operations, called a sketch, from a natural language question, then specializes the sketch into a concrete query program with mentions of entities, attributes and relations, and finally executes the program against the database. This pipeline decomposition approach enables us to train the system by leveraging out-of-domain data and semi-synthetic data generated by GPT-3, thus reducing overfitting and shortcut learning even with limited amount of in-domain training data. Our code can be found at `https://github.com/PaulDrm/DISCOSQA`.

1 Introduction

Space debris are uncontrolled artificial objects in space that are left in orbit during either normal operations or due to malfunctions. Collisions involving space debris can generate secondary debris which can cause more collisions, potentially leading to a runaway effect known as Kessler Syndrome (Kessler and Cour-Palais, 1978; Kessler et al., 2010), which in the worst-case scenario could make large ranges of orbits unusable for space operations for multiple generations.

Therefore, space agencies have established departments responsible for cataloging the space debris environment, which can be used for space traffic management, collision avoidance, re-entry
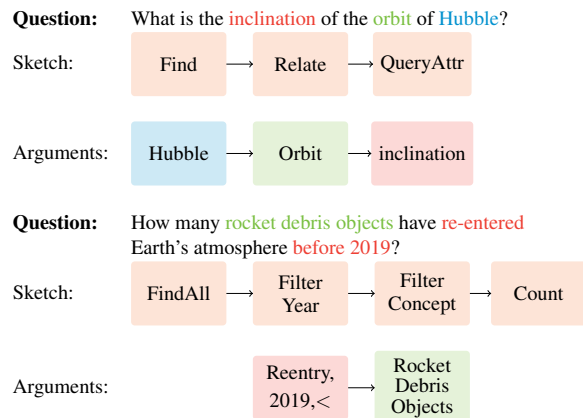

Figure 1: Two representative queries for DISCOS and their decomposition according to the Program Induction method.

analysis, and raising public awareness of the problem.[1]

The European Space Agency (ESA) has catalogued over 40,000 trackable and unidentified objects in its DISCOS (Database and Information System Characterizing Objects in Space) Knowledge Base (KB) (Klinkrad, 1991; Flohrer et al., 2013). Accessing this information efficiently often requires technical expertise in query languages and familiarity with the specific schema of DISCOS, which may fall outside the skillset of the engineers searching for relevant information in the database. In this project, we developed a question answering system for the DISCOS KB. This deployed prototype enables ESA engineers to query the database with complex natural language (English) questions, improving their ability to make informed decisions regarding space debris.

Recent breakthroughs in open question answering have been achieved using large language models that have been fine-tuned as dialog assistants, such as ChatGPT.[2] These models, however, are

---

[1] `https://tinyurl.com/44tc24d4`
[2] `https://chat.openai.com`

black boxes that store knowledge implicitly in their parameters which makes it hard to guarantee that their answers are supported by explicit evidence, understand their failures and update them when the supporting facts change. In contrast, parsing a question into a query program and then executing it on an explicit KB is guaranteed to provide a factual correct answer provided the KB and query program are correct. Our approach is particularly useful for applications such as satellite operations where accuracy and reliability are critical.

The main challenge for this project was that no training set or example questions were available for the DISCOS KB. This issue, combined with the large amount of unique and diverse objects in the database, precluded a straightforward application of common supervised learning techniques. Although possible strategies for solving this task, such as direct semantic parsing of the query with seq2seq models, were identified in the literature, they suffer from problems with compositional generalization (Herzig et al., 2021; Furrer et al., 2020). Furthermore, very little work has been done on generalizing to KB element components that were never seen during training (Cao et al., 2022b; Das et al., 2021a; Huang et al., 2021).

To overcome these challenges, we apply and adapt a methodology from the literature called Program Transfer (Cao et al., 2022b) to significantly reduce the required dataset for adequate generalization over the complete DISCOS KB. This is a two-step approach. For each user query first a program sketch is predicted, consisting of a sequence of query functions where the arguments are either variables or placeholders, then the representation of the query is compared to the representations of the KB entities, in order to fill out the placeholders with arguments relevant to the query text. The underlying query language of this approach is called Knowledge-oriented-Programming-Language (KoPL) for which two representative example questions are shown together with their decomposition into sketch and arguments in Figure 1.

We also conduct a data collection study with domain experts, and we apply a data augmentation pipeline leveraging the underlying ontology of the KB and prompting a Large Language Model (LLM) to generate automatically more training examples. The architecture was retrained with different domain-specific LMs and baselines to deter-

mine the benefits of using a domain-specific pre-trained encoder.

The main contributions of this paper are:

- Applying and adapting a methodology described in the literature for complex knowledge base question answering (CKBQA) on a novel industry-relevant database, with a large and dynamic set of unique entities;
- Collecting a new dataset on this database from domain-experts and leveraging the in-context learning capability of LLMs for data augmentation on it;
- Evaluating the use of domain-specific LMs as different encoders on our curated dataset;
- Demonstrating the effectiveness of the approach by achieving comparable results to general-purpose LLMs

2 Related Work

Low-resource CKBQA Pre-trained language models have demonstrated state-of-the-art performance in semantic parsing for complex question answering on KBs where the same logic compounds are contained in both the training and validation sets (Furrer et al., 2020). However, they struggle with compositional generalization, where the "compounds" (combinations) of components are diverse between training and validation, even if all components (entity, relation, program filters) have been seen during training (Herzig et al., 2021). Das et al. (2021b) explored retrieval-based methods to pick the top n similar examples from the training set and use them as additional input for the prediction. In theory, this would make it possible to reason over changes on the KB by only adding new examples to the training set without the need of retraining the whole model. Another approach is adapting the architecture of language models to incorporate the structure of a KB directly for the prediction. For example, Huang et al. (2021) ranked FreeBase KB entities by using an EleasticSearch search API to identify these entities. When generating the query program, instead of entities a special token is predicted, which in the post-processing step get replaced by the top ranked entity identified by ElasticSearch. Although, achieving good results, it is unclear how this would translate to queries with multiple entities and also has the typical limitations of Elastic-Search. Another method is the Program Induction and Program Transfer method, where a sequence

of *functions*, or a *sketch*, is generated from the input query. A single function here stands for a basic logic operation on the KB. The premise is that the sketch is mostly dependent on the formulation of the input query and less dependent on the KB, therefore the training on a source domain can transfer to inference on a target domain. In a second step, the particular inputs that each function in the sketch receives are identified from the elements of the KB through comparing their representations with the one of the model at the specific function. During training, the goal is to create sophisticated representations for the components of the KB as well as for the query that can generalize to components which were not seen during training (Cao et al., 2022b).

Domain-specific Language Models Using self-supervised pre-trained language models is the de-facto standard approach in modern natural language processing (NLP). These models are trained on large volumes of text, learning representations that can generalize over natural language variations and capture long-term dependencies between the input tokens. During fine-tuning, these learned features and representations commonly lead to improved results on the downstream task. While the interplay between the amount of in-domain data, model capacity and training regime is complex (Zhao et al., 2022), as a general rule, training these models on in-domain task-related text improves the performance of this task (Maheshwari et al., 2021; Berquand et al., 2021; Arnold et al., 2022; Joshi et al., 2023). An alternative approach involves modifying the pre-training objective according to the domain. In the context of question-answering with tabular data, it was explored how a language model could function during pre-training as a SQL-query excecutor, predicting the results of an automated created SQL query on the corresponding concatenated table, to elicit an understanding of the underlining dependencies in tables. This approach resulted in improved performance on related downstream tasks (Liu et al., 2022). For KBs, together with the standard Masked-Language-Modelling loss an architecture called Kepler was tested that also minimizes a contrastive loss on related KB triples, where both the correct triples and randomly perturbed incorrect triples are scored, and the loss penalizes scoring the perturbed triples higher than the correct ones (Wang et al., 2021a).

3 Dataset Collection and Use

We used two sources of data for the study. The first one is the original dataset which first introduced the KoPL-format for general-domain knowledge-base question answering (described in §3.1). The second one is for our particular domain with information about objects in space (§3.2). We also describe in §3.3 how we further collected training data for fine-tuning a language model, and how we augmented this dataset with new question-answer pairs (§3.4).

3.1 The KQA Pro Dataset

The KQA pro dataset (Cao et al., 2022a) is a large scale dataset for complex question answering over a knowledge base. It contains over 120,000 diverse questions for an subset of entities from the Freebase KB and their associated relations, attributes and concepts. The reasoning process to arrive at a solution is provided in the form of a Knowledge-oriented-Programming-Language (KoPL), which was designed specifically for this dataset. The question-program pairs were automatically generated by randomly sampling the extracted KB and using novel compositional templates to create a canonical form of the question and associated answer. To increase ambiguity these questions were then paraphrased and controlled by Amazon Mechanical Turk workers (Cao et al., 2022a).

3.2 The DISCOS Database

The ESA Database and Information System Characterising Objects in Space (DISCOS)[3] is a regularly updated source for information about launches, launch vehicles, objects, spacecraft registration numbers, mission-specific information (mass, mission objective, operator), and most importantly orbital and fragmentation data histories for all trackable as well as unclassified objects. With currently over 40,000 objects being tracked, this tracking provides rich information for ESA offices monitoring and managing space debris, collision avoidance, re-entry analyses, and contingency support. Other actors, such as research institutes, government organisations, or industrial companies from ESA Member states can apply for an account to access the information provided by DISCOS free of charge. A comparison between the the DIS-

---

[3] https://discosweb.esoc.esa.int/

COS KB and the KB used for the KQA Pro dataset can be seen in Table 1

Dataset	#Entities	#Relations	#Concepts
KQA Pro	16,960	363	794
DISCOS	73,354	32	39

Table 1: Entity, relation, and concept counts for the KQA Pro and DISCOS KBs

3.3 Data Collection

As there was no question-program-answer training set available on the DISCOS KB, potential queries had to be collected from domain-experts via a simple user interface. The interface allowed domain experts to input queries of interest, along with their username and feedback, see Appendix E. Based on the domain experts' feedback, a manually labeled baseline dataset of around 102 question-KoPL program pairs was created.

3.4 Data Augmentation

To extend the limited baseline dataset and increase diversity of potential queries, we augmented the dataset by creating paraphrases of the questions, shown to add robustness to question answering systems (Fader et al., 2013; Narayan et al., 2016). For each unique program sketch, the schema of the ontology was used to alter the arguments of the single functions in that sketch. For example, for the program *Find('Saturn V')* → *QueryAttr('mass')*, the concept of the *Find* function argument (*Saturn V*) was identified as *LaunchVehicle*. Subsequently, the ontology was queried for other entities of the same concepts and also for associated attributes to substitute the argument for the *QueryAttr* function.

In order to generate appropriate questions for each "augmented" program, we used the few-shot and in-context learning capabilities of GPT-3 (Brown et al., 2020). A prompt was curated, consisting of question-program pairs from the manually labeled dataset, so that GPT-3 could generate a question for an unlabelled augmented program. The sampling included all programs from the manual dataset with the same sketch as the augmented one, as well as using examples with the same relation type, which showed great abilities to generate a correct question. Additionally, an instruction section was added to the prompt, consisting mainly of a list of expansions for acronyms commonly used in the KB's ontology. This ensured

that acronyms were expanded correctly and reduced the likelihood of hallucinations by the LLM for these acronyms. Examples of generated questions with their corresponding programs can be found in Appendix D. The prompt schema can be found in Appendix A.

The benefits of an automatically generated dataset include cost-effective data sample generation, while also ensuring a balanced distribution of complex and simple queries as well as common (ex: Saturn V, Hubble Space Telescope) and uncommon arguments (ex: L-186, PSLV-Q). With the use of LLMs, the generated questions also have already slightly different semantics and syntactic structure as the generation process is inherently statistical and can be adjusted with parameters such as the temperature. LLMs can also leverage their stored world knowledge, e.g. we observed the entity name " 天舟四号 " to be automatically translated into the english "Tianzhou 4" designation. More examples can be seen in Appendix D. However, it is important to note that the question generation is not foolproof and could be further optimized e.g. through additional prompt engineering or a subsequent data cleaning procedure.

4 Methodology and Model Architecture

We describe in §4.1 the problem that our model aims to solve. In §4.2, we describe what modification we applied to the methodology from (Cao et al., 2022b).

4.1 Problem Definition

The task is defined in the following way, given a natural language question Q we want to predict a program y that traverses the knowledge base K and produces an answer A for Q. This means:

$$A = y(Q, K), \quad K = \{E, R, C, A\},$$

where E, R, C, A represent respectively the mutually disjoint sets of entities, relations, concepts and attributes in K. More specifically for a set entities in the training set E_t, the task is to be able to generalise to the set of E_v, which were unseen during training, with $E = E_t \cup E_v$, $E_t \cap E_v = \emptyset$. Therefore, we apply a program induction and transfer methodology, predicting for Q a program, a tuple of actions,

$$y(Q, K) = (o^1(arg^1), \ldots, o^t(arg^t)),$$

$$o^i \in O, \forall i = 1, \ldots, t; \quad arg^t \in E \cup R \cup C \cup A,$$

where O is a set predefined basic functions executable on the KB, with each of them taking one disjoint set as inputs from the pool of arguments from the KB. In the first step, the sketch $[o^1, \ldots, o^t] \in O^t$ is generated by encoding the question Q with a pre-trained language model and using its representation as the starting point for a GRU-based decoder with attention mechanism (Bahdanau et al., 2015; Cao et al., 2022b).

The input arguments for each function, $o^1, \ldots, o^t$, are chosen in a second step by calculating the probability between the encoded representations of the i-th candidate R_i^t in the KB at position t in the sequence, and the representation of the decoder at position t, h^t. The probability is computed as:

$$p(arg^t | R_i^t, h^t) = \text{softmax}(R_i^t, h^t),$$

and is followed by choosing the most likely candidate as the input argument for the function.

We refer the reader to Cao et al. (2022b) for the details. In addition to the standard BERT model we also train RoBERTA-based (Liu et al., 2019) domain-specific encoders (see Appendix B).

4.2 Enhancements to Program Transfer

At the beginning, the prediction for entities, relations, and operations ($\{<, >, =\}$) had to be re-implemented as it was not available in the associated source code.[4]

The standard methodology of comparing the representations to all inputs during training posed another challenge for entities, as during the training step the gradients for all them would need to be stored, which were exceeding the available virtual random access memory on the system.

As a result, instead of comparing the representation at each sequence position to all entities in the KB, only the subset of entities in the current batch are compared with each other.[5] In addition, random samples from the complete entity set where selected and added to the batch to also give signals to entities not occurring in the training set.

More formally, let $X^{(j)}$ be the jth batch from the training set, E be the overall set of entities, and $X_E^{(j)}$ be the set of entities of the training batch. At each training batch j, we use the set of entities $e^{(j)}$

to compare the probability of their representations against the candidate input argument in batch j, as defined by:

$$e^{(j)} = X_E^{(j)} \cup f^{(j)}, \quad (1)$$

where $f^{(j)} = \{e_1, e_2, \ldots, e_n\}$ are n randomly sampled entities from E without replacement. Furthermore, for extracting time values from queries we use the SUTime library (Chang and Manning, 2012), whereas for extracting numerical patterns we use a simple regex schema. In addition, for parsing the predicted inputs into a form than can be read by the KoPL engine a parser function had to be written as well as a method for assigning the dependencies between the single basic functions. Other small adaptation of the original approach included the addition of a normalization layer between the linear layers for the prediction of the single arguments as well as masking-out padding tokens from the loss calculation of the function generation component. A schematic overview of the pipeline can be seen in Appendix C.

5 Experiments

We next describe our dataset (§5.1), the way we trained our models (§5.2) and our results (§5.3).

5.1 The DISCOS-Questions-Programs Dataset

The creation of the training and validation datasets involved the following steps: First, all the unique sequences of functions were extracted from the manually labeled dataset. Then, ten augmented programs were generated for each unique program in the manually labeled dataset by substituting the inputs and generating questions as described in the data augmentation methodology. It is noteworthy that the entities included in the manually labeled dataset were not considered as candidates for the data augmentation. The questions for the augmented examples were generated using the large language model *code-davinci-002*.[6] Through trial and error, a temperature of 0.75 was used for generation as it produced diverse examples while still capturing the meaning of the associated KoPL-program. Finally, the augmented dataset was split into training and validation datasets, where the original manually labeled dataset and 0.05% of randomly sampled examples from the augmented

---

[4] https://github.com/thu-keg/programtransfer

[5] Note that these entities are part of the training set entities, hence are all known apriori.

[6] https://platform.openai.com/playground

dataset were used as the validation set and the rest of the augmented dataset was used for training. In a subsequent filtering step, programs that appeared in the validation set were filtered from the training dataset. The resulting DISCOS-Questions-Programs (DQP) dataset consists of 905 samples for training and 151 samples for validation.

5.2 Model Training

The preliminary results indicated that training directly on the DQP dataset did not lead to convergence in entity prediction. We then proceeded to first train on the out-of-domain original KQA Pro dataset and then further train on the DQP dataset. The hyperparameters for pretraining on the KQA pro dataset were adopted from the original paper. For training the DQP dataset, the hyperparameters were left unchanged, except for an adapted learning rate for the decoder, which was set to 10^{-4} instead of 10^{-3}. For the experiments, different domain-adapted models were used as the encoder and then compared to each other as well as baseline models. For more information about the domain-adapted models see Appendix B.

5.3 Results

The analysis of different models was challenging as multiple components (functions, entities, relations, etc.) need to be predicted to arrive at the full program that can be run against the KB. Therefore, the analysis was divided into two parts. Firstly, the accuracy at the lowest validation loss for each component was compared separately between all the different trained models, providing an overview of each model's best predictive performance for each component. The results can be seen in Table 2. Although no model consistently outperformed the others on all components, CosmicRoBERTa (Berquand et al., 2021) achieved the highest performance on four out of six accuracies.

The validation loss curves showed that during training, the validation accuracy can drop for one component while it rises for another. The validation loss curves can be found in Appendix F.

For deployment of a single model, it is necessary to identify a checkpoint where the model predicts accurately across all components. To obtain a more holistic view of the performance, the models were also compared by summing up the normalized validation losses for each component.

The results of summing up the normalized validation losses with equal weights are shown in Ta-

Accuracy	BERT	Kepler	RoBERTa	CR
Function	0.797	*0.812*	0.796	**0.826**
Entity	0.874	0.887	*0.907*	**0.927**
Attribute	**0.955**	*0.948*	*0.948*	0.948
Relation	0.938	*0.983*	*0.983*	**1**
Concept	0.872	*0.896*	**0.92**	0.89
Operations	**1**	**1**	**1**	**1**

Table 2: Accuracy at lowest validation loss for each respective component. CR stands for CosmicRoBERTa. The best score for each component is highlighted in **bold**, second best in *italics*.

ble 3. On average, the RoBERTa-base model has the lowest validation loss, followed by the Kepler model. However, no model consistently outperforms the others in terms of prediction accuracy. From an application perspective, the most important metrics are the accuracy in predicting functions and entities. To obtain the correct answer, the most crucial step is to predict the correct sequence of functions, and for multi-hop queries, it is essential to identify the correct starting entity. In addition, the number of unique entities is magnitudes higher than the number of unique attributes or relations in the KB, which makes identifying the right entity more difficult. Among the models considered, CosmicRoBERTa stands out as the model that performs well both in predicting functions and entities. Unfortunately, the Kepler model only sporadically showed improvements over RoBERTa-base. The reason for this could be the very limited pre-training corpus, which as a result was significantly smaller than the one from CosmicRoBERTa As a result, for the purpose of deploying a single model, the decision was made to choose CosmicRoBERTa.

Accuracy	BERT	Kepler	RoBERTa	CR
Min. sum valid loss	0.21	*0.171*	**0.11**	0.252
Function	**0.79**	0.734	0.775	*0.789*
Entity	0.874	0.887	*0.894*	**0.927**
Attribute	0.935	**0.948**	*0.941*	0.915
Relation	**0.978**	0.95	*0.969*	*0.969*
Concept	0.853	0.8841	*0.908*	**0.927**
Operations	**1**	*0.97*	**1**	0.94

Table 3: Accuracy at the lowest validation loss summed over all compoennts. The best score for each component is highlighted in **bold**, second best in *italics*. CR stands for CosmicRoBERTa.

The results of the prediction are overall very impressive, with high accuracy scores over all com-

ponents. It is especially worth highlighting that from over 40,000 entities in the database, only 400 appear in the training and validation set and only 1 of 71 entities in the validation set also appear in the training set. Despite this, some models achieve an accuracy of over 90% in predicting entities, demonstrating their strong ability to generalize to entities not seen during training, which was a critical user requirement for our system.

In addition, we benchmarked our method to recently released general purpose models such as ChatGPT[7] and GPT-4 (OpenAI, 2023). For each model, training set examples were randomly added to the prompt until the respective model's context limit is reached. Then the models were prompted to generate the right program for a question from the validation set. Our methodology has an overall accuracy of predicting the right program completely of 48%, which is higher than the performance of around 25% of ChatGPT and comparable with the performance of GPT-4 of around 50%. A detailed comparison can be found in Table 4. This further demonstrates the efficiency of our methodology as it can be run at a fraction of the necessary compute as well as locally on consumer-grade hardware.

	CosmicRoBERTa	GPT-4	ChatGPT-3.5
Functions	**0.79**	**0.79**	0.516
Entities	**0.93**	0.86	0.41
Relations	**0.97**	0.87	0.32
Concepts	**0.93**	0.82	0.61
Attributes	**0.92**	0.84	0.72
Overall	0.48	**0.5**	0.25

Table 4: Accuracy of deployed model (CosmicRoBERTa) versus general purpose models ChatGPT-3.5 and GPT-4. The best score for each component is highlighted in **bold**.

6 Conclusion

We developed a system for ESA to address the challenge of answering complex natural language questions on their DISCOS KB. The main obstacles included a lack of training data, the diversity and regular updates of the database entries, and the need for an economically feasible solution. The program transfer for complex KBQA methodology was selected for its potential to reduce the amount of required training samples through transfer learning and its capability to potentially gener-

alize for examples which were never seen during training. A data collection study was conducted with domain experts, which was then used to augment the data through leveraging the underlying ontology of the KB and prompting a large language model to generate fitting questions. The architecture was retrained with different domain-specific models and baselines to determine the benefits of using a domain-specific pre-trained encoder. Although the results were mixed, the best performance was achieved by CosmicRoBERTa, a pre-trained model on a space domain corpus. With an accuracy of over 90% of predicting the right entity on the validation set over the vast pool of candidate entities, the method demonstrates its strong ability to predict the correct input arguments for unseen examples. This is further demonstrated in the comparison with general-purpose models such as ChatGPT or GPT-4, where our method achieved competitive results. Therefore, this approach has the potential to be extended to other databases and query languages in the future, especially in scenarios where there are few to no training examples.

Limitations

The study has several clear limitations. Firstly, the training and validation datasets used in this study are still relatively small. A larger dataset would give more robust results for comparing different encoders. Additionally, the experiments were only conducted on the ability to generalize to unseen entities and not on the ability to generalize to unseen sketch types, which is also of key importance when addressing low resource CKBQA.

Moreover, the methodology used in this study relies on annotated question-program pairs, which are expensive to collect. Learning only from question-answer pairs or even a question with an indicated difficulty based on whether the model was able to answer the question, could be more easier to collect. While the models achieve high accuracy on most of the knowledge base components, overfitting can occur at different stages during training, leading to high accuracy for one component at one training step but poor accuracy for another component at another step. In the future, revising the training procedure or the model setup may help address this issue.

---

[7]https://openai.com/blog/chatgpt

Ethics Statement

In any safety-critical context like spacecraft operations, there is an inherent risk associated with the use of automatic methods supporting human operators. The transparency of the predicted programs could mitigate this issue as it allows even for an engineer with limited knowledge about the underlying query method to interpret the program to some degree. In any case, the developed systems might support human analysis and decision making by decreasing workload, but cannot replace it. As mentioned before the DISCOS KB can be accessed after creating a user account. We plan on publishing the created question-program paris and trained models online in accordance with ESA's guidelines.

Acknowledgments

References

Alexandre Arnold, Fares Ernez, Catherine Kobus, and Marion-Cécile Martin. 2022. Extraction d'informations de messages aéronautiques (NO-TAMs) avec des modèles de langue appris de façon auto-supervisée (information extraction from aeronautical messages). In *Actes de la 29e Conférence sur le Traitement Automatique des Langues Naturelles. Volume 1 : conférence principale*, pages 335–344, Avignon, France. ATALA.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Audrey Berquand, Paul Darm, and Annalisa Riccardi. 2021. Spacetransformers: Language modeling for space systems. *IEEE Access*, 9:133111–133122.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022a. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6101–6119, Dublin, Ireland. Association for Computational Linguistics.

Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022b. Program transfer for answering complex questions over knowledge bases. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8128–8140, Dublin, Ireland. Association for Computational Linguistics.

Angel X. Chang and Christopher Manning. 2012. SU-Time: A library for recognizing and normalizing time expressions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3735–3740, Istanbul, Turkey. European Language Resources Association (ELRA).

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021a. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen

Tan, Lazaros Polymenakos, and Andrew McCallum. 2021b. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618, Sofia, Bulgaria. Association for Computational Linguistics.

T Flohrer, S Lemmens, B Bastida Virgili, H Krag, H Klinkrad, E Parrilla, N Sanchez, J Oliveira, and F Pina. 2013. Discos-current status and future developments. In *Proceedings of the 6th European Conference on Space Debris*, volume 723, pages 38–44.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking compositional generalization in pre-trained models using intermediate representations. *ArXiv preprint*, abs/2104.07478.

Xin Huang, Jung-Jae Kim, and Bowei Zou. 2021. Unseen entity handling in complex question answering over knowledge base via language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 547–557, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Harshit Joshi, Abishai Ebenezer, José Cambronero, Sumit Gulwani, Aditya Kanade, Vu Le, Ivan Radiček, and Gust Verbruggen. 2023. Flame: A small language model for spreadsheet formulas.

Donald J. Kessler and Burton G. Cour-Palais. 1978. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research: Space Physics*, 83(A6):2637–2646.

Donald J Kessler, Nicholas L Johnson, JC Liou, and Mark Matney. 2010. The kessler syndrome: implications to future space operations. *Advances in the Astronautical Sciences*, 137(8):2010.

H. Klinkrad. 1991. Discos - esa's database and information system characterising objects in space. *Advances in Space Research - ADV SPACE RES*, 11:43–52.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: table pre-training via learning a neural SQL executor. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *ArXiv preprint*, abs/1907.11692.

Himanshu Maheshwari, Bhavyajeet Singh, and Vasudeva Varma. 2021. SciBERT sentence representation for citation context classification. In *Proceedings of the Second Workshop on Scholarly Document Processing*, pages 130–133, Online. Association for Computational Linguistics.

Shashi Narayan, Siva Reddy, and Shay B. Cohen. 2016. Paraphrase generation from latent-variable PCFGs for semantic parsing. In *Proceedings of the 9th International Natural Language Generation conference*, pages 153–162, Edinburgh, UK. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021a. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021b. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Zheng Zhao, Yftah Ziser, and Shay B Cohen. 2022. Understanding domain learning in language models through subpopulation analysis. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*.

A Instructions for Generating Questions

> *"Here is a list of knowledge graph query programs in JSON format, each with its corresponding question in English. Acronyms are expanded with the following dictionary: ["GEO": "Geostationary Orbit", "IGO":"Inclined Geosynchronous Orbit", "EGO":"Extended Geostationary Orbit", "NSO":"Navigation Satellites Orbit", "GTO":"GEO Transfer Orbit", "MEO":"Medium Earth Orbit", "GHO":"GEO-superGEO Crossing Orbits", "LEO":"Low Earth Orbit", "HAO":"High Altitude Earth Orbit", "MGO":"MEO-GEO Crossing Orbits", "HEO":"Highly Eccentric Earth Orbit", "LMO":"LEO-MEO Crossing Orbits", "UFO":"Undefined Orbit","ESO":"Escape Orbits"] Program: {program} Question: {question} (Repeated until maximum prompt limit is reached)*

B Domain-specific Language Models

CosmicRoBERTa A domain-specific language model from the SpaceTransformer family trained with basic masked-language-modelling on a corpus consisting of 75M words (Berquand et al., 2021).[8]

Kepler We trained our own domain-specific language model by appending the KB pre-training objective as described by Wang et al. (2021b). The in-domain text data for the Knowledge-augmented LM was specifically mined to be closely related to the topic of DISCOS. To achieve this, we collected scientific papers about space debris and documents from ESA about their internal mission operation procedures. Additionally, we added the online available dataset from SpaceTransformers[9] to the training data.

The complete dataset comprises approximately 17.6 million words, with around 70% used for training and the remainder for validation. To predict triples, we converted our database into a set of (*head*, *relation*, *tail*) triples, where *head* and *tail* are entities represented by their English name or

description, and *relation* is a relation represented by a unique token for the relation type.

The KG triplet datasets consist of approximately 640,000 triples, which are split into 636,000 for training, 2,000 for validation, and 2,000 for testing. These triples represent approximately 59,000 entities and 32 relations. We trained our model using the available Kepler implementation.[10]
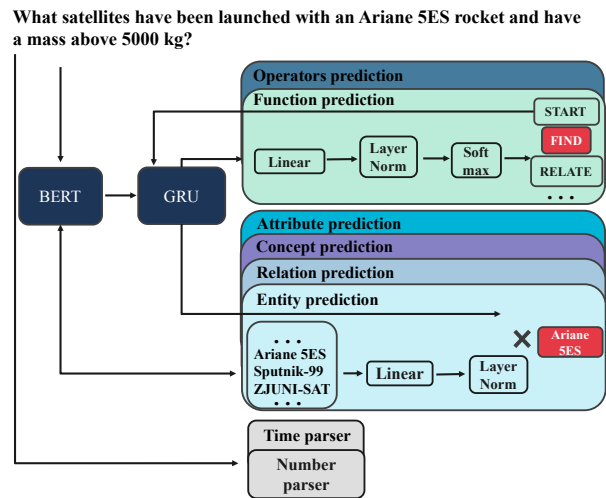
C Pipeline Components

Figure 2: Overview over components in query processing pipeline.

D Example Augmented Programs and Generated Questions

```
[{"function":"FindAll", "inputs":[], "dependencies":[]},
{"function":"FilterConcept", "inputs":["LMO"], "dependencies":[0]},
{"function":"Relate", "inputs":["orbit"], "dependencies":[1]},
{"function":"FilterNum", "inputs":["depth", "0.3", "="], "dependencies":[2]},
{"function":"FilterConcept", "inputs":["UnknownObjClass"], "dependencies":[3]},
{"function":"Count", "inputs":[], "dependencies":[4]}]
```

Question: How many objects are there that are currently in the LEO-MEO crossing orbits and are classified as unknown and have a depth of 0.3 m?

```
{"function":"Find", "inputs":["State Remote Sensing Center"], "dependencies":[]},
{"function":"Relate", "inputs":["host_country"], "dependencies":[0]},
{"function":"FilterConcept", "inputs":["Entity"], "dependencies":[1]},
{"function":"What", "inputs":[], "dependencies":[2]}]
```

Question: "Which operators are based in the host country of the State Remote Sensing Center?"

```
{"function":"FindAll", "inputs":[], "dependencies":[]},
{"function":"FilterDate", "inputs":["epoch", "2022-04-08", "="], "dependencies":[0]},
{"function": "FilterConcept", "inputs":["Launch"], "dependencies":[1]},
{"function": "Count", "inputs":[], "dependencies":[2]}]
```

Question: "How many launches are planned for 8th of April 2022?"[a]

---
[a]Interestingly the generated question implies that 8th of April 2022 lies in the future, which is in accordance with OpenAI's training set cut-off in September 2021

E   Interface for Data Collection and Access to System

For implementing a simple user interface, the popular Python library Streamlit was used. Besides providing the input query the user can also provide feedback, which potentially could be used to improve the model by identifying right or wrong answers and adding them to the training set. Another button allows the user to generate an answer for a question from the validation dataset to get a better feeling of what questions could be answered.

Query DiscosWeb database with natural language

Enter user name for feedback purpose

Note: Try to not use keywords, but full-fledged questions.

Enter a query for the data base

How many payloads with a mass above 100 kg have reentered after 2010?

| Run | Random question |

Additional comment for feedback
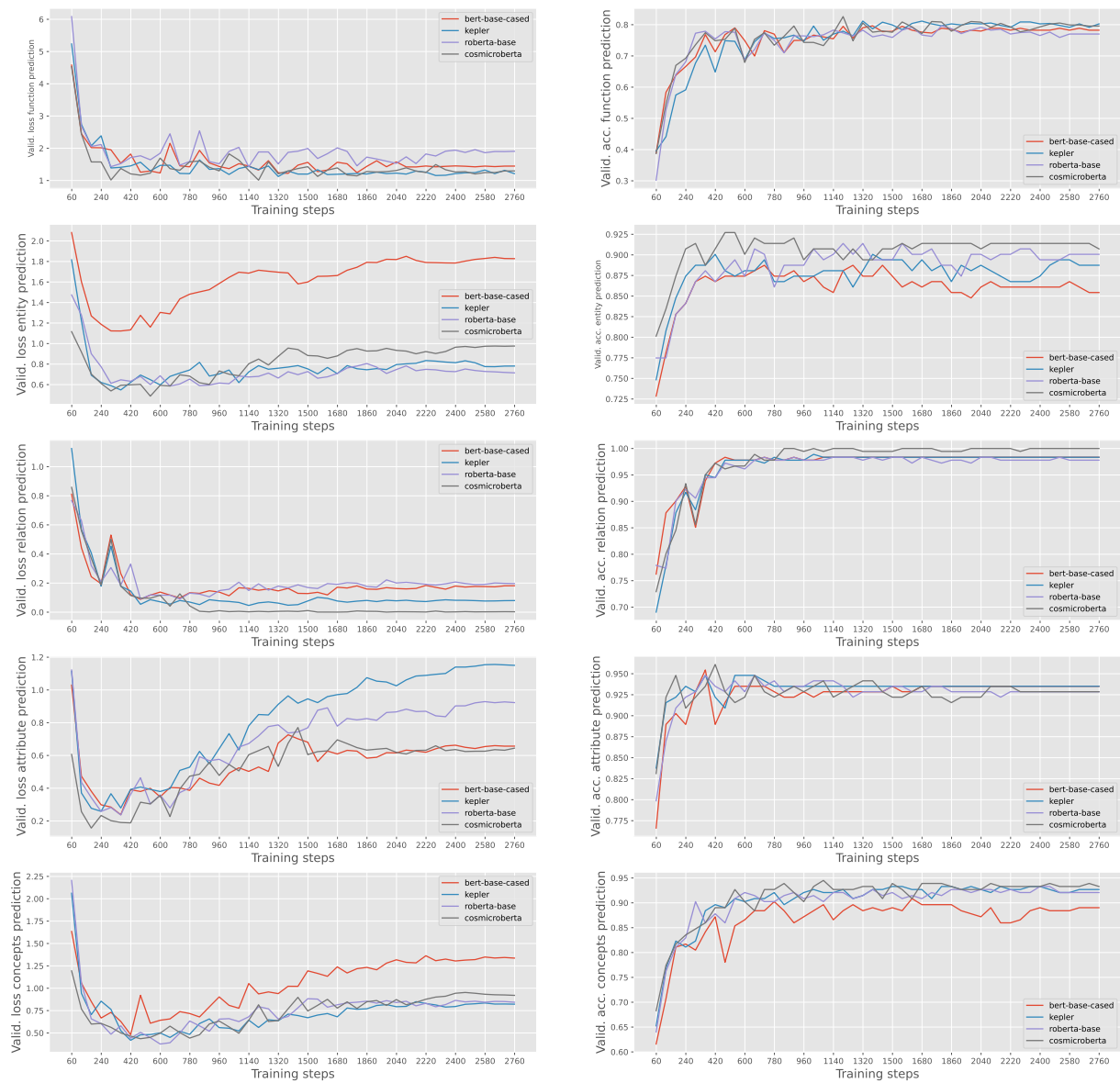
👍 👎

F   Validation Loss Curves

Figure 3: Validation loss and Accuracy over number of training batches for each component and each tested model.