Adapting Foundation Models for Operator Data Analytics

Manikanta Kotaru
Microsoft
mkotaru@microsoft.com

Abstract

The complexity of operator networks and myriad of specialized metrics produced by network function providers present a formidable challenge in retrieving and analyzing operator data, a vital component for network operations. This necessitates specialist intervention, which is time-consuming and limits its customization. This paper proposes Data Intelligence for Operators Copilot, a natural language interface for retrieval and analytics tasks on operator data, leveraging foundation models. It addresses the challenges posed by operator data through a novel application of semantic search to effectively provide necessary context regarding specialized metrics. The system has outperformed state-of-the-art natural language interfaces for databases, when applied to an operator-specific benchmark dataset of expert-generated representative queries, with 66% execution accuracy.

CCS Concepts

• **Information systems** → **Information retrieval**; • **Networks** → **Mobile networks**; **Network management**; **Network monitoring**; • **Computing methodologies** → **Natural language processing**.

Keywords

Foundation models, 5G, Operator networks, Data retrieval

ACM Reference Format:
Manikanta Kotaru. 2023. Adapting Foundation Models for Operator Data Analytics. In *The 22nd ACM Workshop on Hot Topics in Networks (HotNets '23), November 28–29, 2023, Cambridge, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3626111.3628191

1 Introduction

Telecommunications operators gather vast amounts of data, which includes node-level, gNodeB-level, user-level, and flow-level data. This data is utilized for various purposes, such as network monitoring, tracking Key Performance Indicators, node management, network capacity management, network planning, charging, policy enforcement, consumption trend analysis for different types of traffic, and debugging. In commercial operator networks, the number of such counters and metrics that are regularly computed, often exceeds several thousands accounting for tens of Gbps of data transfer [27, 29]. Retrieving relevant metrics and visualizing them is crucial for network operations. However, the complexity of modern wireless systems and the vast number of counters involved make this task challenging, necessitating expert knowledge to perform this essential operation.

The process today involves specialists, with expert knowledge, creating dashboards for a limited number of metrics, which the operators browse through to obtain relevant information. However, if operators require customized data, such as visualizing throughput for a specific user rather than aggregate throughput, or need access to different set of metrics for complex debugging purposes, a loop through the specialists is required. The specialists would need to identify the relevant variables, write code in database query language to combine them in an appropriate manner, create and share a dashboard.

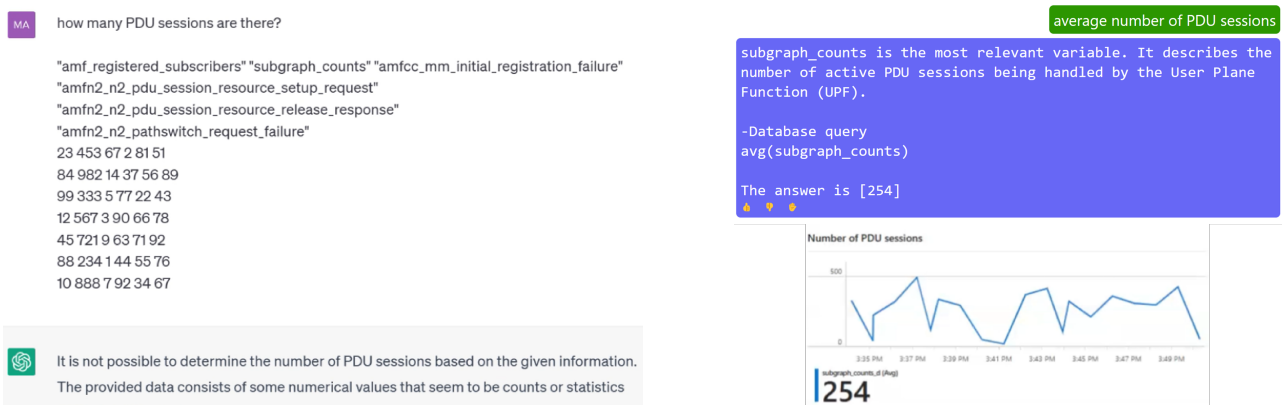
This paper poses the following question: *can the operators interact with their data by asking simple questions in natural language, without having to remember any of the complex counter names or how to combine them in a database query language?* Such a system has potential to significantly transform the status quo by providing a more natural way to interact with operator data without heavy reliance from specialists, reducing the time to mitigate network issues and providing more value from the operator data by reducing the barrier to gain customized insights.

Recent advancements in natural language processing have led to the development of foundation models [2] such as GPT-4 [30], which have greatly improved the creation of natural language interfaces for data interaction. Systems built on these foundation models have demonstrated state-of-the-art performance on standard text-to-SQL datasets [33]. However, despite their success, these models face challenges when it comes to specialized and niche domains like operator data. Specifically, there are three main challenges that hinder their performance in this domain:

- **Specialized information:** The data counter names and metric definitions are highly specialized and specific to the operator. This information may not even be publicly available for metrics produced by a few network function

HotNets '23, November 28–29, 2023, Cambridge, MA, USA

ACM ISBN 979-8-4007-0415-4/23/11.
https://doi.org/10.1145/3626111.3628191



(a) ChatGPT **(b) DIO copilot**

Figure 1: Comparison of responses between ChatGPT and DIO copilot for a sample question on operator data

vendors. This causes the foundation models to miss relevant text patterns and struggle to understand the fields and how they relate to the user question.

- **Huge data:** The number of data counters and metrics is massive. It is infeasible to provide in-context description in sufficient detail within the prompt size limits of even the most advanced foundation models [30].
- **Numerical accuracy:** While the foundation models excel at logical reasoning, they are not adept at providing numerically accurate answers [25]. However, such accuracy is vital for decision making in network operations.

This paper proposes to overcome these challenges by incorporating four components to enable Data Intelligence for Operators Copilot (hereby referred to as DIO copilot):

- **Domain-specific database:** The system incorporates comprehensive textual descriptions of various metrics. It also contains a few complex and bespoke functions that operate on these metrics to obtain entities that are of interest. The inclusion of such domain-specific information enhances the understanding of users' questions and operator data by the foundation models. Contrarily, many contemporary 'natural-language to database-query-language' systems do not integrate such comprehensive context, and only include database schema, as the foundation models exhibit good understanding of the data fields [33].
- **Semantic search:** The system utilizes a foundation model to discern the necessary metrics to effectively answer a user's question. Given the constraints on prompt size, the system employs a context extractor to sift through the domain-specific database, extracting metrics whose descriptions have a high semantic proximity to the user's query. This approach manages the complexity of huge number of metrics by effectively filtering only a small number of metrics and their descriptions that fit within the limited prompt size of foundation models.
- **Few-shot learning:** To ensure numerical accuracy, the system utilizes foundation models' code generation capabilities for data retrieval and analytics, instead of directly prompting with data. Few-shot learning using a select number of expert-generated example user questions and corresponding query language code, pertaining to operator data, significantly enhances the accuracy of the code generated.
- **Expert feedback:** The paper introduces a novel method enabling users to solicit expert assistance. This mechanism triggers the creation of a GitHub [11] issue that domain experts can address by contributing to the domain-specific database, fostering a system that improves with usage.

A preliminary analysis has been conducted to demonstrate the validity of the approach. The system has been implemented and evaluated using more than 3000 metrics produced by a major virtual network function provider for 5G core related to key network functions. The system incorporates GPT-4 foundation model. A benchmark dataset consisting of 200 expert-generated questions and corresponding reference answers, representative of questions in commercial operator deployments, has been created. DIO copilot outperformed state-of-the-art natural language interfaces for databases, when applied to operator data, demonstrating an execution accuracy [33] of 66% compared to 48% achieved by state-of-the-art adapted to work with operator-specific benchmark dataset.

1.1 Contributions

A natural language interface for retrieval and analytics of operator data is enabled by two main contributions: (1) Novel application of semantic search in 'natural language to database query language' systems to provide a curated context of specialized metrics without overwhelming the limited prompt size of foundation models, and (2) A mechanism to enhance the system with usage through incorporation of expert feedback and contribution, achieved by re-purposing repository issues. The paper further highlights research challenges and opportunities in enabling data intelligence for operators.

2 Related work

The field of enabling natural language interfaces to database queries has attracted significant interest due to its potential in facilitating non-expert data analysis [3, 5, 10, 34, 43–45, 47]. Recent advances in foundation large language models (LLMs) [6, 30, 49], with powerful zero-shot reasoning and domain generalization capabilities, have led to state-of-the-art performance [33] on benchmark text-to-SQL datasets [46].

However, these systems have limitations in terms of out-of-domain generalization [23, 40]. [32, 38] have demonstrated the effectiveness of similarity-based demonstration retrieval in such cross-domain settings. These approaches append user question with annotated examples selected based on relevance to user question. [35] demonstrated that adapting Codex [30] to a specific domain by prompting it with few annotated examples can be more effective than fine-tuning a smaller language model on the same examples. However, these approaches are complementary to our work as they focus on annotated training examples alone, akin to few-shot learning, and do not directly address the specialized nature of counter names and metric definitions of operator data. Extending our system using such complimentary techniques [32, 33, 38, 42] is part of future work. High quality benchmark datasets [24, 46, 50], including domain-specific datasets [7, 16, 48], have been instrumental to progress in this field. This paper develops a domain-specific dataset for operator data.

Developing systems for supporting and analyzing big data analytics for cellular network data [8, 13–15, 18, 19] is an active area of research. [22] presents a persuasive argument for the potential benefits of foundational models to the domain of networking by drawing similarities between networking tasks and similar counterparts in natural language processing. However, challenges in enabling natural language interfaces to operator data remain largely unexplored.

3 System design

Figure 2 describes the overall architecture of the DIO copilot . Prior to delving into the specifics, the following provides a concise summary of the operational procedure of the system. The copilot has access to logs and databases containing operators' data. When a user types a question in the message bar as shown in Figure 1b, the system provides the metrics that are most relevant to answering the user question, provides a description of what the metric measures, details a database query that it would run using the data corresponding to relevant metrics, and provide a numerically accurate answer to the user's data retrieval and analytics query. Further, a dashboard for visualization of time-series data of relevant metrics is generated. The response includes options for feedback as well as a button to request expert contributions. The system comprises of four main components:

3.1 Domain-specific database

Foundation models have revolutionized data interaction by enabling natural language interfaces. However, these models face challenges when dealing with 5G operator data, particularly in understanding and providing relevant answers to data retrieval and analytics queries. For example, as illustrated in Figure 1a when queried about the number of PDU sessions using representative operator data and metric names, these models fail to produce relevant answers and struggle to comprehend the associated fields.

One of the key challenges with specialized information is the translation and understanding of complex and often product-specific metrics. Detailed documentation of each of these metrics provide invaluable context to foundation models and enhance the model's understanding of specific fields. For example, the documentation of 'amfcc_n1_auth_request' refers to 'The number of authentication requests sent by AMF. The AUTHENTICATION REQUEST message is defined in section 8.2.1 of 3GPP TS 24.501. 64-bit counter'. Each counter within the system is accompanied by an explanation of the specific measurements and the data format. A domain-specific database is built where each text sample contains the definition and description of a particular metric.

Additionally, the domain-specific database features 'function definitions'. Sometimes, it is not straightforward to amalgamate various counters to compute a specific outcome; such a process might necessitate specialist-crafted functions or queries. Hence, we have incorporated a provision that allows specialists to add such bespoke functions and definitions to the system. Function definitions consist of the function name, description of what the function performs, executable function along with a description of the inputs and the outputs.

While one might consider fine-tuning [9, 26, 37] the foundation model using the available documentation, this approach proves to be suboptimal due to the volatility of the documentation. As new metrics and functions are continually added or modified, the process of finetuning the entire model becomes cumbersome, time-consuming, and expensive. Furthermore, the product-specific nature of these metrics could lead to misinterpretations, given that the foundation model is trained across diverse data sets. For instance, terms like 'subgraph_counts' could have varying meanings in different contexts that the model is trained on. Therefore, relying solely on fine-tuning could lead to potential confusion.

To overcome these challenges, a prompt engineering approach [1, 4, 28, 39, 51], that integrates the domain-specific database as additional context in the prompts, has been adopted. By integrating the domain-specific information into the prompt, we equip the model with the necessary knowledge to interpret and respond accurately to queries concerning operator data. Such an approach allows for dynamic updates to the metrics and functions and resolves the ambiguity in the metric names by grounding with respect to product-specific databases.

However, the task is complicated by the substantial number of counters present in a typical commercial operator network. The limited input prompt size of foundational models poses a constraint, as it becomes difficult to accommodate all counter definitions within this space. For example, although GPT-4

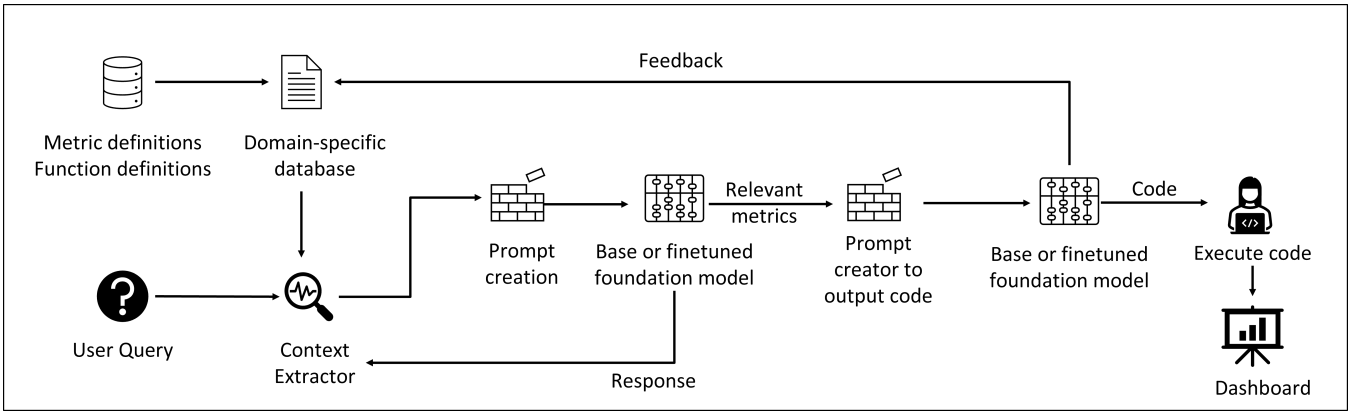


Figure 2: System architecture

can accommodate up to 32,000 tokens, roughly equivalent to 24,000 words [30], the number of counters generated by commercial network function providers [29] exceeds 6,000. Consequently, describing these counters adequately within the input prompt becomes infeasible.

3.2 Extraction of relevant metrics

To overcome this challenge, we employ a context extractor technique that narrows down the metrics in the context to only those that possess semantic proximity to the user's query. This is achieved through the conversion of domain-specific databases and user queries into word embeddings, followed by the application of similarity search to identify the vectors in the domain-specific database that exhibit the closest resemblance to the user question. The model is then prompted to identify the most pertinent metrics from this filtered set in order to address the user's query effectively.

The first step involves an offline process of converting the text samples in domain-specific database into word embeddings. Word embeddings are essentially mathematical representations of word sequences, where words with similar meanings occupy proximate spaces in a high-dimensional vector space. Each of the queries input by the user is converted to word embedding as well. Cosine similarity between the embedding vector of the query and vectors corresponding to the domain-specific database is performed. The text samples corresponding to the highest cosine similarity are identified as the closest to the user question semantically.

After filtering the metrics through the context extractor, we then utilize the foundation model to determine the most relevant metrics to respond to the user's query. The descriptions of all the filtered metrics output from the context extractor are fed as supplemental context to user question, and the foundation model is prompted to identify the metrics in the context that are most relevant to answering the user question. Since only the definitions of top few semantically close metrics are considered, their descriptions do not overwhelm the input prompt of the foundation models. The foundation model is a critical component in our workflow, leveraging their named entity recognition and natural language understanding capabilities to select the most pertinent metrics among those filtered by the context extractor.

3.3 Data processing

Foundation models have demonstrated capabilities to answer data analytics queries on unstructured data. However, doing so directly on operator data not only incurs a high cost, but it can also result in numerical inaccuracies and be infeasible due to data volume. Instead, an approach leveraging the code generation capabilities of the foundation models is adopted.

The most relevant metrics obtained from the previous step along with the user question are input to the foundation model, which is then prompted to generate database query language code to generate the answer for the user's query and generate code for creating time-series visualization of the relevant variables on a dashboard. Generating code automatically can be a challenging task, often resulting in errors due to the complexity of the query language. To overcome this hurdle, we utilize few-shot learning techniques, enabling foundation models to learn from a limited set of examples and generalize to generate accurate code. Specifically, a few example codes for answering a sample of different user queries are included in the prompt to the foundation model. By exposing the model to carefully curated examples, we enhance its ability to generate error-free code consistently. The generated code is executed on the database in a sandboxed environment [12]. Similarly, few-shot training examples for dashboard generation are used to generate and then execute code for dashboard creation.

3.4 Expert feedback

This paper acknowledges that the foundation model based systems do not always provide accurate and relevant answers [2, 30]. Further, for complicated debugging processes, it is not immediately apparent which metrics to use and how to process those metrics to answer user queries.

Hence, DIO copilot includes mechanisms for incorporating expert feedback and data contributions. Upon receiving a response, the user can optionally request expert assistance

by clicking a designated raised-hand button, which will create a GitHub repository issue. This issue will contain the question, context, and response, and can be resolved through contributions from domain experts. Currently, only a select few pre-identified experts can resolve these issues. The expert data obtained through this process is then added to the domain-specific database and attributed to the relevant expert as its source.

The attribution of responses to the expert authors ensures that experts receive recognition for their contributions, and creates accountability to the information that is being added to the domain-specific database. While limiting the number of experts may create a bottleneck, leading to delays in resolving the issues, the system leaves the possibility to expand the pool of experts or adopting a voting mechanism, similar to Stack Overflow [31], as part of future work.

4 Evaluation

DIO copilot has been built on representative operator counters and metrics produced by a major virtual network function provider for 5G core. The data consists of more than 3000 metrics and statistics across a wide range of essential network functions - Access and Mobility Management Function (AMF), Session Management Function (SMF), NF Repository Function (NRF), Non-3GPP Inter-Working Function (N3IWF), Network Slice Selection Function (NSSF), and User Plane Function (UPF). In commercial operator networks, these counters are then used to derive different entities, monitored through dashboard panels. For example, statistics for AMF call control service measuring 'Initial registration procedure attempts' and 'Initial registration procedure success' are used to produce a dashboard panel listing the 'initial registration procedure success rate'.

The text from the documentation for different metrics, made available by the vNF provider, is extracted and segmented into text samples containing the names and detailed description of each of the counters. The sentence-BERT all-MiniLM-L6-v2 model [36] is used for embedding the text samples and user queries into word-embedding vectors. The FAISS [17] library is used for efficient storing of the embedding vectors in domain-specific database, and for computing cosine similarity. The top 29 most similar text samples are appended as supplemental context to the user query. Few-shot learning is enabled by feeding into the prompt an additional 20 expert-generated tuples consisting of user query, corresponding context, relevant metrics and the PromQL query that generates the correct output when executed.

The resulting prompt, generated using LangChain [21] library, is fed into the GPT-4 [30] foundation model. Maximum number of output tokens is set to 1000 and temperature parameter of the foundation model is set to 0 for repeatable answers to the same query. The same foundation model has been employed to generate a PromQL query to answer user's question and dashboards with time-series data of the relevant metrics. The PromQL language is chosen as it is popular with operator deployments. A basic implementation is available at [20], with the operator-specific data and metrics omitted.

4.1 Benchmark dataset

To evaluate the performance of DIO copilot for data retrieval and analytics tasks on operator data, a benchmark dataset of 200 expert-generated user questions and corresponding reference PromQL expressions is obtained. Each reference response consists of the metrics that are essential to answer the corresponding user question, a PromQL query and a numeric answer. The numeric answer is obtained by executing the reference query on a database comprising synthetic yet representative data for different metrics. None of the training questions used for few-shot learning are incorporated into the benchmark dataset. The queries span an extensive spectrum of metrics related to diverse network functions, and target multiple tasks like retrieval, averaging, sum and rate, and contain up-to three metrics in a single expression. The benchmark dataset is used only for evaluation purposes and no training or finetuning is performed using the dataset.

4.2 End-to-end evaluation

A preliminary analysis has been performed using the benchmark dataset, described in Sec. 4.1, to demonstrate the validity of the proposed approach.

4.2.1 Compared approaches The following approaches and DIO copilot are probed with the same set of test queries from the benchmark dataset.

- **DIN-SQL:** DIN-SQL [33] achieved the state-of-the-art on standard text-to-SQL datasets. The system builds on LLMs like GPT-4 through prompting approaches and decomposes the text-to-SQL task into smaller sub-tasks. Few-shot learning prompt described in the paper [33] is used as a comparison approach. However, there are two modifications. First, rather than using SQL queries, the same set of few-shot learning examples, *i.e.*, user questions and corresponding PromQL queries used as part of DIO copilot , are used as few-shot learning examples. Second, since the entire database schema does not fit in the limited context window of GPT-4 model, approximately 600 of the metric names, that are selected in a uniformly random manner among all the metrics, are provided in the prompt.
- **GPT-4:** Foundation models like GPT-4 have been trained on vast corpus of web data, and have demonstrated significant capabilities to understand the schema of unstructured data and directly perform analytics tasks [30, 35]. The same subset of metrics used in DIN-SQL prompt are used in the prompt of this approach as well, without any examples.

4.2.2 Metric of Merit **Execution accuracy (EX)**, which measures the percentage of times an approach produced an answer that is numerically matching the reference answer, is used to measure the performance of the different approaches.

Approach	EX (%)
DIO copilot	66
DIN-SQL	48
GPT-4	12

(a) End-to-end comparison

LLM	EX (%)
GPT-4	66
GPT3.5-turbo	46
text-curie-001	13

(b) Foundation model

Figure 3: Analysis

4.2.3 Results Table 3a demonstrates that DIO copilot outperforms state-of-the-art approaches when applied to operator data. This can be mainly attributed to supplemental context provided by the context extractor describing the metrics that are semantically closest to user query. The other approaches do not consider such curated context and can only access subset of all the metric names due to limited input size for GPT-4. The small frequency with which metrics used in operator networks are discussed, if at all, in publicly available web corpus and the ambiguity of the terms across multiple domains makes it challenging for the foundation models to understand the field names correctly without the context. For example, for a benmark question on 'LCS NI-LR procedure success rate', DIN-SQL incorrectly evaluates '100 * sum(amfcc lcs ni lr success) / sum(amfcc lcs ni lr attempt)' while DIO copilot correctly identifies 'amfcc lcs network induced location request success' and corresponding 'attempt' metrics as the relevant metrics. Using just the base foundation model, GPT-4, without few-shot learning performs poorly signifying the importance of curated context and few-shot learning examples when working with operators' data.

4.2.4 Impact of the foundation model Different foundation models can be used and each of these models exhibit distinct capabilities, computational costs, and complexities. We considered three foundation models in the form of GPT-4, GPT-3.5-turbo, and Text-curie-001 [30] models. The results from evaluating different foundation models while keeping the rest of the architecture the same are showcased in Table 3b. GPT-4 provides the best performance and there is a significant drop even when the model is switched to GPT-3.5-turbo model. However, it is interesting to note that even the least performing model still outperforms using GPT-4 alone in Table 3a underscoring the importance of context.

4.2.5 Inference cost The average cost of each user query is 4.25 cents as the context information is appended to the query. When GPT-4 model is replaced with GPT-3.5-turbo (see Table 3b), the average cost reduces to 0.35 cents without significant reduction in performance. However, the inference cost is expected to reduce in the future with the advancements in foundation LLMs and inference techniques [41].

5 Opportunities and challenges

This paper demonstrates the viability of an end-to-end system for enabling a natural language interface for operator data. However, to achieve a comprehensive system that significantly simplifies network operations, several challenges must be addressed, necessitating further research.

5.1 Diverse network function vendor formats

The evolution of cellular networks from traditional hardware-based infrastructure to software-defined frameworks has led to a surge in virtualized network function vendors, each presenting unique data formats and semantics. This transformation has introduced significant challenges in integrating data across multiple vendors. Solutions lie in establishing industry-wide data formatting and interoperability standards, as well as leveraging advancements in machine learning for multi-source data integration, semantic understanding, and consistency enforcement. The ability of foundation models to comprehend unstructured data and perform few-shot learning could significantly aid the integration process

5.2 Infusing domain knowledge

By incorporating further domain knowledge, the copilot could better respond to user queries, identify metric relationships, and provide more in-depth network issue insights, potentially improving network management, troubleshooting, and predictive modeling. However, embedding such domain knowledge into the copilot poses significant challenges. Comprehensive domain knowledge in the field of wireless networks includes complex, dispersed and often proprietary knowledge, including vendor-specific metrics, their interdependencies and impact on network performance.

5.3 Network-specific embedding model

Generic embedding models may not fully comprehend the wireless network domain's unique jargon and semantics, potentially affecting system accuracy. A potential remedy could be a wireless network-specific word embedding model, trained on a large telecommunications corpus. This specialized model could better grasp the domain's specific terminology and context, enhancing its ability to identify semantic proximity between user queries and database entries. However, creating such a model involves curating a vast, diverse, and evolving corpus compatible with embedding model generators.

5.4 Correctness and safety

Inaccuracies in copilots' responses could result in erroneous decisions. Further, safety concerns arise, when the copilot interacts with operational databases, such as the risk of unintentional execution of harmful code and controlling access to sensitive data. These challenges, common in AI systems, warrant significant research attention.

Acknowledgments

The author thanks the anonymous reviewers, Victor Bahl, Bozidar Radunovic, Matthew Balkwill, Mustafa Kasap, Sudeep Chakravartty, Ranveer Chandra, Microsoft Azure for Operators Office of the CTO and Networking Research teams for their valuable feedback and insightful discussions.

References

[1] Simran Arora, Avanika Narayan, Mayee F Chen, Laurel J Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. 2022. Ask Me Anything: A simple strategy for prompting language models. *arXiv preprint arXiv:2210.02441* (2022).

[2] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[3] Ruichu Cai, Boyan Xu, Xiaoyan Yang, Zhenjie Zhang, Zijian Li, and Zhihao Liang. 2017. An encoder-decoder framework translating natural language to database queries. *arXiv preprint arXiv:1711.06061* (2017).

[4] Maximillian Chen, Alexandros Papangelis, Chenyang Tao, Seokhwan Kim, Andy Rosenbaum, Yang Liu, Zhou Yu, and Dilek Hakkani-Tur. 2023. PLACES: Prompting language models for social conversation synthesis. *arXiv preprint arXiv:2302.03269* (2023).

[5] Zui Chen, Zihui Gu, Lei Cao, Ju Fan, Sam Madden, and Nan Tang. 2023. Symphony: Towards natural language query answering over multi-modal data lakes. In *Conference on Innovative Data Systems Research, CIDR*. 8–151.

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

[7] Deborah A Dahl, Madeleine Bates, Michael K Brown, William M Fisher, Kate Hunicke-Smith, David S Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

[8] Hong-Ning Dai, Raymond Chi-Wing Wong, Hao Wang, Zibin Zheng, and Athanasios V Vasilakos. 2019. Big data analytics for large-scale wireless networks: Challenges and opportunities. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 1–36.

[9] Shizhe Diao, Ruijia Xu, Hongjin Su, Yilei Jiang, Yan Song, and Tong Zhang. 2021. Taming pre-trained language models with n-gram representations for low-resource domain adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 3336–3349.

[10] Han Fu, Chang Liu, Bin Wu, Feifei Li, Jian Tan, and Jianling Sun. 2023. CatSQL: Towards Real World Natural Language to SQL Applications. *Proceedings of the VLDB Endowment* 16, 6 (2023), 1534–1547.

[11] GitHub. 2023. GitHub Issues. https://docs.github.com/en/issues.

[12] Ian Goldberg, David Wagner, Randi Thomas, Eric A Brewer, et al. 1996. A secure environment for untrusted helper applications: Confining the wily hacker. In *Proceedings of the 1996 USENIX Security Symposium*, Vol. 19. USENIX Association Berkeley.

[13] Ying He, Fei Richard Yu, Nan Zhao, Hongxi Yin, Haipeng Yao, and Robert C Qiu. 2016. Big data analytics in mobile cellular networks. *IEEE access* 4 (2016), 1985–1996.

[14] Anand Iyer, Li Erran Li, and Ion Stoica. 2015. Celliq: Real-time cellular network analytics at scale. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*. 309–322.

[15] Anand Padmanabha Iyer, Li Erran Li, and Ion Stoica. 2017. Automating diagnosis of cellular radio access network problems. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. 79–87.

[16] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760* (2017).

[17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[18] Ilyas Alper Karatepe and Engin Zeydan. 2014. Anomaly Detection In Cellular Network Data Using Big Data Analytics. In *European Wireless 2014; 20th European Wireless Conference*. 1–5.

[19] Mirza Golam Kibria, Kien Nguyen, Gabriel Porto Villardi, Ou Zhao, Kentaro Ishizu, and Fumihide Kojima. 2018. Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks. *IEEE access* 6 (2018), 32328–32338.

[20] Manikanta Kotaru. 2023. Code repository for the paper 'Adapting foundation models for operator data analytics'. Retrieved Nov, 2023 from https://github.com/mkotaru/operator-data-analytics

[21] LangChain. 2023. LangChain. https://github.com/hwchase17/langchain.

[22] Franck Le, Mudhakar Srivatsa, Raghu Ganti, and Vyas Sekar. 2022. Rethinking data-driven networking with foundation models: challenges and opportunities. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. 188–197.

[23] Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. *arXiv preprint arXiv:2106.11455* (2021).

[24] Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, et al. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *arXiv preprint arXiv:2305.03111* (2023).

[25] Tiedong Liu and Bryan Kian Hsiang Low. 2023. Goat: Fine-tuned LLaMA Outperforms GPT-4 on Arithmetic Tasks. *arXiv preprint arXiv:2305.14201* (2023).

[26] Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Domain adaptation with BERT-based domain classification and data selection. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*. 76–83.

[27] McKinsey. 2023. A future for mobile operators: The keys to successful reinvention. https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/a-future-for-mobile-operators-the-keys-to-successful-reinvention.

[28] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. 2023. Augmented Language Models: a Survey. *arXiv preprint arXiv:2302.07842* (2023).

[29] Microsoft. 2023. Azure Operator Insights. https://azure.microsoft.com/en-us/products/operator-insights.

[30] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[31] Stack Overflow. 2023. Stack Overflow. https://stackoverflow.com/.

[32] Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. 2022. Synchromesh: Reliable code generation from pre-trained language models. *arXiv preprint arXiv:2201.11227* (2022).

[33] Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *arXiv preprint arXiv:2304.11015* (2023).

[34] Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, et al. 2022. A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions. *arXiv preprint arXiv:2208.13629* (2022).

[35] Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498* (2022).

[36] Nils Reimers and Iryna Gurevych. 2019. 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. https://arxiv.org/abs/1908.10084

[37] Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. 2019. Adapt or get left behind: Domain adaptation through bert language model finetuning for aspect-target sentiment classification. *arXiv preprint arXiv:1908.11860* (2019).

[38] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633* (2021).

[39] Zhenwei Shao, Zhou Yu, Meng Wang, and Jun Yu. 2023. Prompting Large Language Models with Answer Heuristics for Knowledge-based Visual Question Answering. *arXiv preprint arXiv:2303.01903* (2023).

[40] Alane Laughlin Suhr, Kenton Lee, Ming-Wei Chang, and Pete Shaw. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. (2020).

[41] Sunyan. 2023. The Economics of Large Language Models. https://sunyan.substack.com/p/the-economics-of-large-language-models.

[42] Immanuel Trummer. 2022. CodexDB: Generating Code for Processing SQL Queries using GPT-3 Codex. *arXiv preprint arXiv:2204.08941* (2022).

[43] Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436* (2017).

[44] Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. SQLizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages* 1, OOPSLA (2017), 1–26.

[45] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. *arXiv preprint arXiv:1804.09769* (2018).

[46] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887* (2018).

[47] John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*. 1050–1055.

[48] John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*. 1050–1055.

[49] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

[50] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103* (2017).

[51] Shuyan Zhou, Uri Alon, Frank F Xu, Zhiruo Wang, Zhengbao Jiang, and Graham Neubig. 2022. Docprompting: Generating code by retrieving the docs. *arXiv preprint arXiv:2207.05987* (2022).