Generating Data Augmentation Queries Using Large Language Models

Christopher Buss[1], Jasmin Mosavi[1], Mikhail Tokarev[1], Arash Termehchy[1,2], David Maier[2] and Stefan Lee[1]

[1]Oregon State University
[2]Portland State University

Abstract

Users often want to augment entities in their datasets with relevant information from external data sources. As many external sources are accessible only via keyword-search interfaces, a user usually has to manually formulate a keyword query that extracts relevant information for each entity. This is challenging as many data sources contain numerous tuples, only a small fraction of which may be relevant. Moreover, different datasets may represent the same information in distinct forms and under different terms. In such cases, it is difficult to formulate a query that precisely retrieves information relevant to a *specific* entity. Current methods for information enrichment mainly rely on resource-intensive manual effort to formulate queries to discover relevant information. However, it is often important for users to get initial answers quickly and without substantial investment in resources (such as human attention). Thus, as an alternative to manually writing mappings from entities to queries, one can learn these mappings progressively by leveraging end users' feedback. We evaluate the use of parameter efficient techniques for leveraging a pretrained large language model (LLM) for this task of online query policy learning.

Keywords

Information Integration, Pre-trained Large Language Models, Online Learning, Query Learning, Machine Learning, AI, and Databases, Applied ML and AI for data management, Heterogeneous and federated DBMS

1. Introduction

There is a recognized need to collect and connect information from a variety of data sources [1]. For example, we have recently worked in a large-scale NIH-funded project to augment the information of biomedical entities by querying other biomedical data sources [2]. The focus of this project is to *repurpose* current drugs to treat the symptoms of diseases for which there is insufficient time or resources to develop new treatments [3]. Biomedical researchers often have some local dataset of available drugs. Given a drug in their dataset, a researcher usually needs to query external data sources to find additional information about the drug.

Due to a lack of access or resources, external information often must be retrieved through querying [4]. Many external datasets are only accessible via query interfaces or APIs. Even with access, it may require too much of a resource (e.g., storage space, time) to download and maintain an up-to-date copy of the external dataset. Thus, information relevant to some local entity must often be gathered on a as-needed basis by querying external data sources.

However, formulating a query that extracts specific information can be troublesome. Different data sources often represent the same concept in distinct forms [5]. Thus, one needs to tailor their query to specific data sources. Figure 1 illustrates a case where users have a local dataset of FDA-approved uses of drugs and would like to query an external data source that contains the off-label uses of those drugs. A drug that is identified by one of its brand names (e.g., *Zoloft*) in *FDA-Approved Uses* is referred to by its generic name (e.g., *Sertraline*) in *Off-Label Uses*. Due to heterogeneities, one may not know how to query for a specific external entity prior to investigating the data in the external source. Consider a biomedical researcher who seeks more information about the drug *Zoloft*. Since they are only aware of the structure and content of their local dataset, they query the external data source for *Zoloft*, but this elicits no results. They try again using a more general description of *Zoloft* (i.e., being a *serotonin reuptake inhibitor*). However, their under-specified query produces many results, most of which are irrelevant. After additional trial-and-error, they find a query that retrieves *Sertraline*. More work is required to then merge the local and external entities into one coherent representation.

Manually querying for specific external entities requires too much work. Continuing our example, if the researcher needs more information for another drug in their local dataset, they will need to repeat the process. Moreover, if they need information from multiple exter-

Joint Workshops at 49th International Conference on Very Large Data Bases (VLDBW'23) — Workshop on LLMs and Databases (LLMDB'23), August 28 - September 1, 2023, Vancouver, Canada
bussch@oregonstate.edu (C. Buss); mousavij@oregonstate.edu (J. Mosavi); tokarevm@oregonstate.edu (M. Tokarev); termehca@oregonstate.edu (A. Termehchy); maier@pdx.edu (D. Maier); leestef@oregonstate.edu (S. Lee)

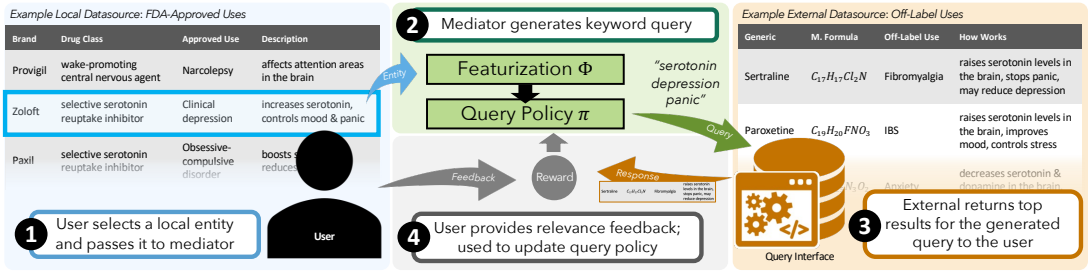


Figure 1: An example of our framework for a single user and single external data source. The user selects (by query, GUI, etc,.) the local entity *Zoloft*. The mediator uses its learned query policy to extract the relevant entity (*Sertraline*) from the external source. The user provides relevance feedback on the results, which is then used to further refine the mediator's querying policy.

nal data sources, then the work required to query for each drug is exacerbated. Furthermore, other researchers with similar information needs must *repeat the same such work themselves.*

To alleviate the burden, one can use a **shared system that automates query formulation**. This **mediator system** acts as a go-between for users and external data sources: a user specifies a local entity (e.g., *Zoloft*) perhaps through a query or a graphical user interface, and the mediator maps the local entity to queries that retrieve the relevant external entities (e.g., *Sertraline*) from their respective external sources.

To the best of our knowledge, such mediators are currently created by *manually writing programs* that generate queries for specific external sources. These programs consist of rules that *cannot* necessarily be reused across data sources. Thus, they require a significant amount of labor and expert attention to build and maintain.

In this paper, we *learn the mediator's query policy online* through user interaction. As illustrated in Figure 1, after the user specifies a local entity, the mediator formulates a query to retrieve records from an external source according to its *query policy* and shows the returned external records to the user. The user then provides feedback on the relevance of the returned records to the local entity. The mediator then uses this feedback to improve its query policy.

Of course, online learning of query policies has its own set of challenges. First, the mediator must learn a sufficiently effective policy in the *short run* so users will continue providing feedback. This challenge is easier to meet when the users' only alternative is tiresome (i.e., manually submitting queries for many local entities) or there are many users providing feedback. Second, the mediator should continue leveraging user feedback to find increasingly effective policies in the *long run* (i.e., it should not be prone to under-fitting to local entities). To help overcome these challenges, we use a pretrained LLM to extract features from local entities and terms. Through pretraining, LLMs encode linguistic knowledge within the rich representations of their outputs. However, to get the most out of an LLM, its output representations should be adjusted to suit the specific task and domain. This is commonly done through finetuning, where the weights of the LLM are trained jointly with the task-specific model. However, finetuning is resource-intensive and may overwrite the LLM's knowledge [6]. Thus, in this paper, we evaluate more parameter efficient techniques for our online setting.

Due to the wide-spread use of keyword query interfaces over external sources, we use an online learning method for formulating keyword queries. We evaluate prefix tuning and attribute encoding as parameter efficient techniques for boosting the performance of an LLM-based query policy learner. We evaluate the techniques using Longformer [7] over four pairs of real-world datasets. We find the techniques may be highly effective for select datasets.

2. Framework

We briefly outline the problem of learning a query policy online. A more detailed discussion of the framework, challenges, and related work can be found in [8]. The **mediator** wraps the local dataset and the query interface over the external data source. We assume the *local dataset* is a single table where each tuple stores information about a distinct entity. We denote the set of local dataset entities as $\mathcal{E}$. Given a local entity e and *external dataset* D, $X(e) \in D$ represents the external entity that is relevant to the local one, where the definition of relevance is domain-dependent. For example, Figure 1 shows excerpts of a local (left) and an external (right) dataset. $\mathcal{E}$ consists of all drugs in *FDA-Approved Uses*. If e is *Zoloft* then the relevant tuple $X(e)$ in *Off-Label Uses* is *Sertraline*.

Given a user-specified local entity $e \in \mathcal{E}$, the mediator must devise and submit a query to the interface to extract $X(e)$. The set of queries accepted by the given interface is $\mathcal{Q}$. In this work, we consider keyword queries. A keyword query q is a string comprised of *terms*. The number of terms in a query is its *length* ℓ. A *querying policy* is a

mapping from local entites to queries $\pi : \mathcal{E} \rightarrow \mathcal{Q}$. Ideally, the policy should produce queries that *effectively* extract external entities relevant to e. One such measure of effectiveness is *reciprocal rank* (RR) $\frac{1}{r}$ where r is the position of the first relevant answer in the results. Continuing our example, given e = *Zoloft*, the mediator must devise a keyword query to extract $X(e)$ = *Sertraline*. One can use the content of the input entity within the output query. However, terms in `Brand` are likely unique to the local dataset. Given this, assume the policy ignores those terms and produces the keyword query q ="*serotonin depression panic*". It submits q to the query interface over the external dataset in Figure 1, which returns the ranked results (*Paroxetine*, *Sertraline*). The RR of this query would thus be $\frac{1}{2}$.

3. LLM-Based Query Learning

Figure 1 illustrates a single *interaction* of online query policy learning. The mediator's policy is refined progressively over many interactions with the objective of maximizing the mean reciprocal rank (MRR) of its queries. As discussed in Section 1, an optimal method would overcome two major obstacles. First, it would maintain user engagement by producing effective queries in the short run. Second, it would have the capacity to improve its policy in the long run.

We use a pretrained LLM in help meet the aforementioned challenges. The model may benefit from the LLM's rich representations of tuples and terms, boosting the model's early performance while also allowing it to fit to the diversity of local entities over time.

Encoding Tuples and Scoring Terms. Given an entity e, we concatenate its terms into a single string s and pass it through an LLM after standard byte-pair-encoding tokenization. The LLM produces a sentence-contextualized representation h_i for each input token. Note that the byte-pair-encoding may break terms into multiple inputs or terms may appear multiple times in the entity, so to produce feature h_i corresponding to term k_i, the output encodings of all these instance are averaged. For convenience, we write this process as: $h_1, ..., h_n = \text{LM}(s)$. These representations capture information about each term given the context of all terms within the entity. However, they lack contextual information about the local data source. Thus, we add this information post-encoding.

We define a feature vector $A_t(k_i, e)$, which contains *distributional* and *schematic* features of terms relative to the local source. One such feature is *Inverse document Frequency* (IDF). Let *Dataset Frequency* (DF) of a term denote the fraction of entities in the local dataset in which the term appears. IDF of a term is the inverse of its DF, and it quantifies how well that term identifies the entity within the dataset. $A_t(k_i, e_t)$ is concatenated onto each corresponding representation forming $c_i = [A_t(k_i, e_t), h_i]$ where $[\cdot, \cdot]$ denotes concatenation. Vector c_i is then passed through a small fully connected layer to predict reciprocal rank r_i for each term.

As discussed in Section 1, we desire parameter efficient methods for adjusting the output of the LLM to our specific task and data. We consider two such methods: *prefix tuning* and *attribute embeddings*.

Prefix Tuning. We use prefix-tuning as an alternative to updating all weights of the LLM [9]. Before passing the base encoding of entity e (i.e., s) through the LLM, we prepend a prompt consisting of d vectors onto s. This contextualizes the output of all tokens in s on this continuous prompt. Feedback is propagated back to these d vectors, resulting in downstream representations that are aligned with our objective.

Attribute Embeddings. To inject the structural information of local entity e within its downstream representation, we adjust the base encoding of s prior to passing it through the LLM [10]. Each attribute (column) within the local dataset is encoded as a vector. These vectors are then added to tokens to provide attribute information. These encodings are updated based on feedback.

Selecting Queries and Updating. To encourage exploration, we apply an ϵ-greedy approach to query formulation [11] — selecting either the next-highest-scoring term or, with probability ϵ, a random term until the desired query length is achieved. User feedback (RR) is used as a prediction target for all query terms appearing in the returned external matches. Unobserved terms have targets of 0 assigned. These term-entity-RR tuples are added to a first-in-first-out buffer of examples for the last 30 observed queries. We train the model by stochastic gradient descent with batches of 8 samples from the buffer at each interaction.

We use a pretrained Longformer model from the Huggingface Transformers library. Parameters are trained using Pytorch's implementation of Adam with default hyper-parameters.

4. Empirical Evaluation

Our datasets are listed in Table 1. Each one contains a local and an external source. We include the entity count and the average number of terms per entity. Each local entity has at least one relevant external entity, but some external sources have additional irrelevant entities that can appear in results. Thus, we also specify the number of relevant external entities. **ChEBI** is derived from sources used in the NIH project discussed in Section 1. The local source uses *DrugBank* data, which contains molecular information about drugs [12]. The external source uses *ChEBI* data, which contains molecular entities used to intervene in the processes of organisms [13]. **WDC** is derived from the English *WDC Product corpus*, contain-

Table 1
Details of datasets used in our evaluation.

dataset	source	avg. terms	#entity	#relevant
Drugs	Local	108	13,725	413
	External	168	46,976	
WDC	Local	67	57,109	55,247
	External	72	55,247	
ChEBI	Local	178	5,483	5,753
	External	73	189,467	
CORD-19	Local	305	250,575	250,575
	External	48	340,826	

ing products scraped from many sites [14]. **CORD-19** contains research records related to COVID-19 [15]. We split CORD-19 into two sources: one containing abstracts (local) and one containing the remaining attributes (external). **Drugs** contains reviews from *Drugs.com* (local) [16] and descriptions of the same drugs in *Wikipedia* (external).

Interactions. We simulate a series of interactions. Each interaction is initiated by sampling a local entity. Given the entity, the mediator generates a query of length ℓ and submits it to the external source, which returns its top-20 results using BM25. The query is then scored based on simulated feedback (i.e., ground truth).

Sampling. Entity preference tends to follow a Zipf distribution $1/i^s$ where i popularity rank and $s \approx 1$ [17]. Thus, users request the $i'th$ most popular entity approximately twice as often as the $(i+1)'th$ most popular entity. We simulate user preference by sampling local entities from a Zipf distribution ($s = 1$). We randomly assign popularity, which is held constant across methods.

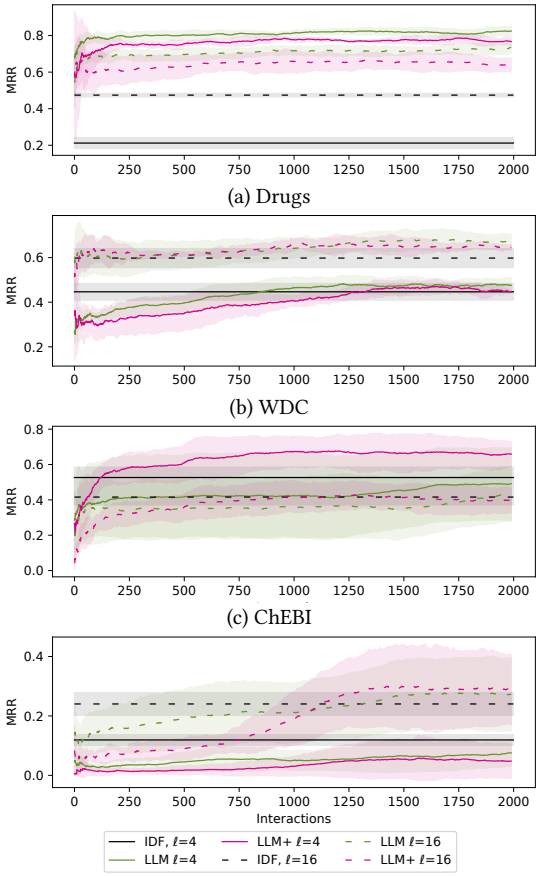
Evaluation Metric. We compute MRR as a sliding average over the previous 500 interactions. We report the average of three runs each comprising 2000 interactions. We plot this average against the current interaction. We include error bands around each line to show a 95% interval for standard error across runs.

Hyperparameters. We treat query length as a hyperparameter and use $\ell \in \{4, 16\}$. We use $d = 5$ prefix tokens for prefix tuning along with a moderate amount of exploration ($\epsilon = 0.05$).

Static IDF. To help contextualize performance, we include a naive policy for comparison. *Static IDF* produces queries using the top-ℓ terms in the content of e based on their IDF. As explained in Section 3, IDF quantifies term specificity within a dataset.

4.1. Results

We seek to understand whether prefix tuning and attribute encoding lead to more effective query policies. Figure 2 compares the LLM-based model with and without prefix tuning and attribute encoding along with Static IDF.



(a) Drugs
(b) WDC
(c) ChEBI
(d) CORD-19

Figure 2: Longformer and IDF comparison for queries of length 4 and 16. LLM+ uses both prefix tuning and attribute encoding whereas LLM uses neither.

Our results indicate that these techniques may drastically help the model for some datasets and keyword lengths. For example, in Figure 2c, we observe *LLM+* $\ell = 4$ exceeding the performance of *LLM* $\ell = 4$ by a large margin. Since ChEBI has 21 attributes in total, it may specifically benefit from the use of attribute encodings. On the other hand, we observe these techniques producing worse results on *CORD-19* and *Drugs*. In contrast to *ChEBI*, the local sources for both *Drugs* and *CORD-19* contain one long textual field with few to no other attributes. Besides the *review* text field, *Drugs* also contains *drugName* and *condition*. Terms from *drugName* tend to be effective and since *drugName* always appears before all other attributes, the positional encodings learned by the pretrained model may be enough to help *LLM* identify terms originating from *drugName*.

Since *CORD-19* contains a single *abstract* text field,

attribute encodings should have little to no effect on performance. Thus, prefix tuning likely degraded the initial performance of *LLM+* $\ell = 4$ in Figure 2d. It is possible that prefix tuning requires more feedback to be effective. If this is true, than it may be possible to balance short run and long run performance by adjusting the amount of parameters within the prefix.

References

[1] N. S. Foundation, N. I. of Health, Smart health and biomedical research in the era of artificial intelligence and advanced data science (sch), 2021. URL: https://www.nsf.gov/pubs/2021/nsf21530/nsf21530.htm.

[2] E. C. Wood, A. K. Glen, L. G. Kvarfordt, F. Womack, L. Acevedo, T. S. Yoon, C. Ma, V. Flores, M. Sinha, Y. Chodpathumwan, A. Termehchy, J. C. Roach, L. Mendoza, A. S. Hoffman, E. W. Deutsch, D. Koslicki, S. A. Ramsey, Rtx-kg2: a system for building a semantically standardized knowledge graph for translational biomedicine, bioRxiv (2021). URL: https://www.biorxiv.org/content/early/2021/11/01/2021.10.17.464747.

[3] T. T. Ashburn, K. B. Thor, Drug repositioning: identifying and developing new uses for existing drugs, Nature Reviews Drug Discovery 3 (2004) 673–683.

[4] P. Wang, R. Shea, J. Wang, E. Wu, Progressive deep web crawling through keyword queries for data enrichment, in: SIGMOD, 2019, p. 229–246.

[5] X. L. Dong, D. Srivastava, Big data integration, PVLDB 6 (2013).

[6] M. McCloskey, N. J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in: Psychology of learning and motivation, volume 24, Elsevier, 1989, pp. 109–165.

[7] I. Beltagy, M. E. Peters, A. Cohan, Longformer: The long-document transformer, arXiv:2004.05150 (2020).

[8] C. Buss, J. Mosavi, M. Tokarev, A. Termehchy, M. David, S. Lee, Effective Entity Augmentation By Querying External Data Sources, Technical Report, 2023. URL: https://web.engr.oregonstate.edu/~termehca/papers/entityarg.pdf.

[9] X. L. Li, P. Liang, Prefix-tuning: Optimizing continuous prompts for generation, 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), 2021, pp. 4582–4597.

[10] P. Dufter, M. Schmitt, H. Schütze, Position information in transformers: An overview, Computational Linguistics 48 (2022) 733–763.

[11] A. Slivkins, Introduction to multi-armed bandits, Found. Trends Mach. Learn. 12 (2019).

[12] D. Wishart, Y. Feunang, A. Guo, E. Lo, A. Marcu, J. Grant, T. Sajed, D. Johnson, C. Li, Z. Sayeeda, et al., Drugbank 5.0: a major update to the drugbank database for 2018, in: Nucleic Acids res. 2017 Nov 8, 2017.

[13] J. Hastings, G. Owen, A. Dekker, M. Ennis, N. Kale, V. Muthukrishnan, S. Turner, N. Swainston, P. Mendes, C. Steinbeck, Chebi in 2016: Improved services and an expanding collection of metabolites, Nucleic acids research 44 (2016) D1214–D1219.

[14] A. Primpeli, R. Peeters, C. Bizer, The wdc training dataset and gold standard for large-scale product matching, in: Companion Proceedings of The 2019 World Wide Web Conference, 2019, pp. 381–386.

[15] L. L. Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Eide, K. Funk, R. M. Kinney, Z. Liu, W. Merrill, P. Mooney, D. Murdick, D. Rishi, J. Sheehan, Z. Shen, B. Stilson, A. D. Wade, K. Wang, C. Wilhelm, B. Xie, D. A. Raymond, D. S. Weld, O. Etzioni, S. Kohlmeier, Cord-19: The covid-19 open research dataset, ArXiv (2020).

[16] F. Gräßer, S. Kallumadi, H. Malberg, S. Zaunseder, Aspect-based sentiment analysis of drug reviews applying cross-domain and cross-data learning, in: International Conference on Digital Health, 2018, pp. 121–125.

[17] C. Cunha, A. Bestavros, M. Crovella, Characteristics of WWW client-based traces, Technical Report, 1995.