Federated Data Science to Break Down Silos [Vision]

Essam Mansour
Concordia University, Canada

Kavitha Srinivas
IBM Research, USA

Katja Hose
Aalborg University, Denmark

ABSTRACT

Similar to Open Data initiatives, data science as a community has launched initiatives for sharing not only data but entire pipelines, derivatives, artifacts, etc. (Open Data Science). However, the few efforts that exist focus on the technical part on how to facilitate sharing, conversion, etc. This vision paper goes a step further and proposes KEK, an open federated data science platform that does not only allow for sharing data science pipelines and their (meta)data but also provides methods for efficient search and, in the ideal case, even allows for combining and defining pipelines across platforms in a federated manner. In doing so, KEK addresses the so far neglected challenge of actually finding artifacts that are semantically related and that can be combined to achieve a certain goal.

1. INTRODUCTION

Open Data initiatives have led to the development of Open Data portals that provide machine-readable and structured datasets in topics, such as health, education, transportation, agriculture, and food. They are driven, for example, by governments, e.g., USA [40], Canada [6], or organizations, such as WHO [45] and WTO [46], and provide access to thousands of datasets. Encouraged by the availability of this data and the FAIR principles [44], data science projects are increasingly striving at making datasets and related data science experimentation automatically and efficiently findable, accessible, interoperable, and reusable. This includes sharing data science pipelines and derived insights, such as code, notebooks, datasets, and technical papers.

Unfortunately, despite artifacts of experimentation and creation of pipelines becoming increasingly more open, most of the artifacts are scattered across various open source repositories, such as GitHub or GitLab. Furthermore, documentation describing the work is available along with code on Jupyter notebooks, blogs in domains, such as Medium, and open repositories of preprints, e.g., ArXiv. Recently, we have therefore seen the rise of initiatives and projects, such as Agora [39], with the goal of providing the foundations of how to technically combine data science pipelines in decentralized and dynamic environments, where data, algorithms, etc. are distributed. While these projects concentrate on the question *how to technically combine artifacts*, they neglect questions, such as *what artifacts should be combined (across platforms, servers, etc.) to achieve a certain goal* and *how do we find artifacts that are semantically similar or connected*. In this vision paper, we are closing this gap by proposing a federated data science platform, called KEK [1], which addresses these neglected questions to break down silos in data science (DS).

Achieving this vision begins with the need to find, combine, and reuse artifacts as they are currently locked away in silos. There is no well-defined way of sharing these artifacts enhanced with semantic descriptions or even general metadata, neither much within a given data science platform and definitely not across multiple platforms. Thus, data scientists cannot automatically find relevant datasets and build a new pipeline on top of related ones since there is no way to identify them. As a practical use case and example, let us consider the case of reproducing experimental results of published articles, and analyzing insights driven from datasets.

Example. The problem is illustrated in Figure 1 – Laboratory 1 has a pipeline in a Java-based machine learning library (MLLib) operating on Dataset 1 to produce insights after enriching Dataset 1 with a local dataset; while Laboratory 2 has a pipeline in a Python machine learning library (Sklearn) that operates on Dataset 2 to produce insights described in a recent paper. At a *semantic* level, Dataset 2 could be joined with Datasets 1 and 3. Similarly, the pipelines are *semantically* equivalent; albeit in different programming languages and libraries. Yet, neither laboratory has any way to understand exactly what has been accomplished in the scientific community with respect to the datasets available at a specific data portal, e.g., Data Portal 1.

Existing data science platforms, such as MLFlow [49] and AutoML [12], tend to expand silos by locking-in pipelines and driven insights with limited or no collaboration support to force scientists to use the same platform. While a number of data science portals already exist, such as OpenML [41] and Kaggle[23], they still expect each user to load all open datasets, pipelines, and insights into their specific platforms – even before users can collaborate. Access to this community effort should not be restricted to a limited set of APIs, as in Kaggle. A more flexible mechanism to allow sharing of **datasets**

[1] KEK is the initials of the authors' first name. Kek means "raiser up of the light" in ancient Egypt.

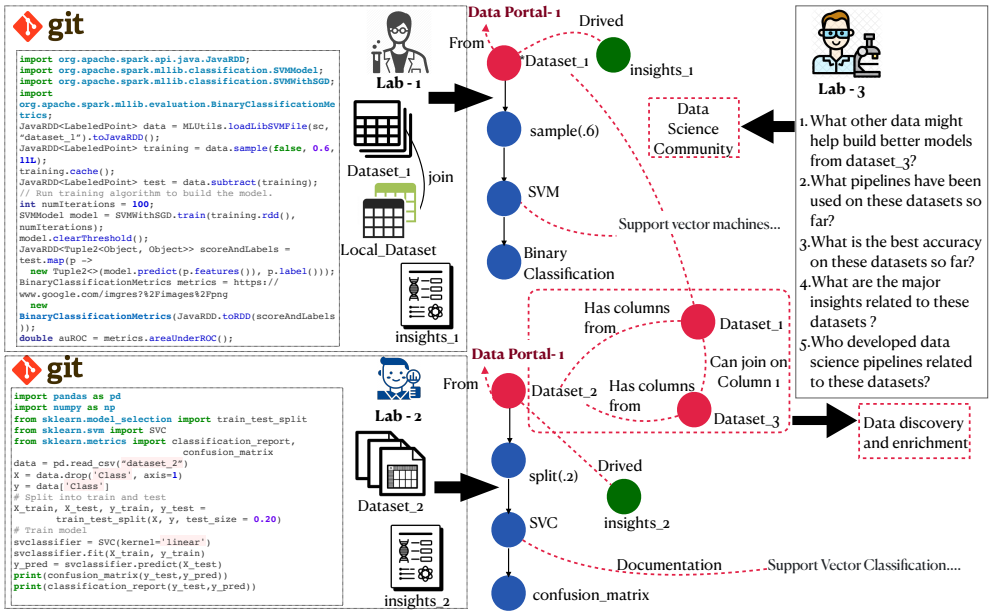

Figure 1: An overview of data science (DS) experimentations suffering from silos of data, pipelines, and insights. These silos prevent communication among the DS community and lead to consuming more time in data preparation, authoring pipelines, and finding insights related to datasets. The required automation to break down silos is denoted in red color.

and their associated **data science** artifacts is needed.

KEK therefore aims to provide a mechanism for the scientific community to discover and learn from each other's work automatically. In particular, KEK will help (i) discover and extract relevant data, (ii) enable scientists to collaborate more effectively regardless of the DS platforms they use, (iii) support efficient discovery of the most recent insights related to a dataset, (iv) enable scientists to reuse and combine (parts of) existing DS pipelines in novel ways, (v) enable reproducibility of experimental results with ease, and (vi) encourage innovative applications to automate several aspects of DS based on the most recent DS experimentation.

One of the key concepts to enable this vision and overcome silos is to abstract from syntactical differences of existing platforms and instead focus on the semantics of datasets, artifacts, and pipelines. Once we understand the semantics, we can more easily identify similar or matching artifacts and combine them in a federated manner. Instead of creating yet another silo by limiting KEK to a non-flexible standard, another key consideration is to retain a maximal degree of flexibility by capturing metadata and semantics in a flexible graph format. In our example from Figure 1, for instance, each laboratory's artifacts (stored in databases, file systems, or from a GitHub repository) are represented and indexed by an abstract graph representation that can be shared with other laboratories as illustrated in Figure 2.

We present an architectural overview of KEK in Section 2. Section 3 discusses how KEK could be used in practice. We discuss the research gaps for reaching our vision in Section 4, and related work in Section 5. Section 6 concludes the paper.

2. THE KEK PLATFORM

KEK aims to break up data silos by extracting and representing semantic information about data and artifacts in a flexible graph structure. The nature of extraction in KEK therefore results in a set of labeled graphs that together form decentralized data science knowledge graphs (**DSKG**s). KEK manages DSKGs using RDF-based knowledge graph technology because (a) it already includes the formalization of rules and metadata using a controlled vocabulary for the labels in the graphs ensuring interoperability, (b) it has built-in notions of modularity in the form of named graphs, so for instance, each laboratory's specific project could get its own named graph, (c) it is schema-agnostic, allowing the platform to support reasoning and semantic manipulation, e.g., adding new labelled edges between equivalent artifacts, as the platform evolves, and (d) it has a powerful query language with federated support (SPARQL) [3].

The KEK platform consists of four main sub-systems, as illustrated in Figure 3, and provides support for federated data science: (i) extracting semantic information from data items (datasets, pipelines, insights, artifacts, etc.), (ii) discovering links and similarities among data items at different granularities, such as datasets, tables, and pipelines, (iii) decoupling the semantics of experimentation on data items (pipelines and insights) from the used data science platform, (iv) interlinking these semantics with the relevant datasets, (v) processing complex queries efficiently in geo-distributed settings, (vi) synchronize the local DSKG with local datasets and scripts of pipelines at scale.

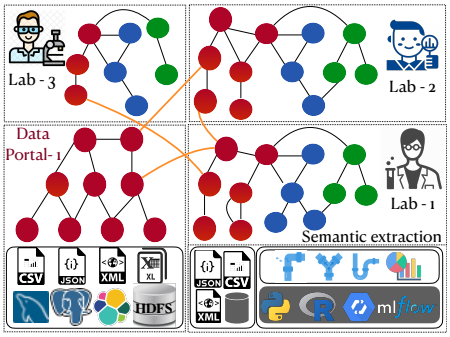

Figure 2: In KEK, decentralized data science knowledge graphs interconnect datasets to relevant pipelines and insights.

DSKG Management. In KEK, the DSKG construction sub-system profiles local datasets to construct a knowledge graph interconnecting data items, e.g., datasets, tables, and columns, accessed locally. The sub-system also maintains DSKG with the semantics captured and extracted from scripts of pipelines and insights. The data owner uses KEK to publish the graph to be accessible via the Web. In KEK, the DSKG services index local datasets and pipelines and maintain up-to-date local graphs capturing the extracted semantics.

KEK Federated Services. KEK provides federated services over geo-distributed DSKGs to allow automatic discovery and learning from data science projects across multiple data science users and heterogeneous data sources. A key feature of these services is to create and maintain links between decentralized DSKGs via, for example, link prediction. Another feature is a query processor that performs federated queries over the local knowledge graph and multiple other KEK portals to help scientists find and join datasets, pipelines, etc.

KEK Interface Services. KEK is designed to support interoperability with existing data science platforms and enable effective communication with data scientists. Thus, KEK provides API libraries to enable different data science platforms to communicate with KEK portals. In addition to structured queries over DSKGs, KEK supports natural language questions that help users easily find answers to their questions and extract the required information directly. A KEK portal is a RESTful server that accepts HTTPS calls.

KEK Foundations. To enable automatic learning from DSKGs, KEK harnesses a broad range of ML approaches including Graph Neural Networks (GNNs) [47] to support different functionalities, such as semantic data enrichment and pipeline automation. Our vision of KEK leverages parallelization and computation sharing to efficiently enable analytical workloads.

3. KEK IN USE

To avoid the dependency to a central instance or authority, KEK is envisioned as a federated platform of independent KEK portals, as shown in Figure 2. Organizations, such as enterprises, countries, or research labs, can then deploy their own instances of a KEK portal on top of their data lake. KEK offers a unique way for organizations to maximize data science potentials by capturing and learning from the usage and interdependencies of their data science artifacts including datasets, pipelines, and derived insights. Researchers, data scientists, and ML engineers, can deploy a KEK portal to capture the semantics of their pipelines and insights and use the KEK functionality to access artifacts shared by remote KEK portals. Hence, the KEK functionality could be implemented by different systems to run on private or public servers. Moreover, cloud providers can provide KEK portals as a service with varying degrees of reliability, performance, and security.

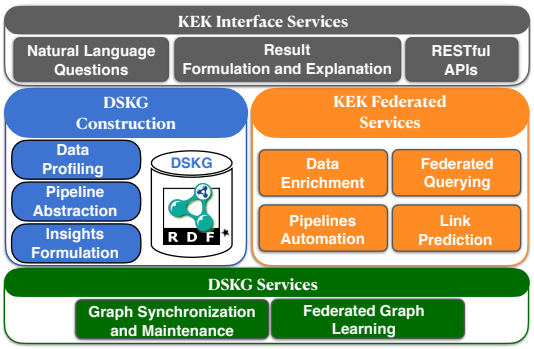

Figure 3: The KEK platform architecture.

Bootstrapping. When a new KEK portal wants to join, the first step is to use the *DSKG Construction* component (Section 4.1) to analyze the locally available data items, capture provenance, etc. and build a local DSKG covering datasets, processes, pipelines, and insights. The next step is to use the KEK *Federated Services* (Section 4.2) to "connect" the local DSKG to the ones from other KEK portals as illustrated in Figure 2.

Maintenance. As data scientists work on their projects and ideas, new datasets, pipelines, insights, etc., are continuously created. Hence, KEK portals need to regularly update their DSKG using the Construction components (Section 4.1) as well as *DSKG Services* (Section 4.3). Since this naturally affects the relationship to data items at other KEK portals, the information about the updates are shared, and the DSKG updated using the KEK Federated Services (Section 4.2).

Users of the KEK Platform. Different types of users interact with the system in different ways using the KEK Interface Services (Section 4.4). An administrator, for instance, might need a slightly different interface than a regular user who might prefer to use a natural language interface. Executing a user request in general can then easily entail using all other KEK components illustrated in Figure 3. As a concrete example, a researcher might want to work with a new dataset. Using the KEK infrastructure, it will be possible to find similar or joinable datasets as well as conclusions derived from similar datasets along with the pipelines that were used in

the process. Hence, given a specific task, users can use KEK to explore and propose potential analyses that have been used in similar cases. For data-driven journalism, given some desired insight, the KEK infrastructure can help find supporting datasets and pipelines.

4. RESEARCH CHALLENGES

This section highlights the open research challenges and opportunities of KEK's components.

4.1 DSKG Construction

In KEK, there is a need for novel methods to capture the semantics of a data science pipeline and its driven insights while interlinking the captured semantics with relevant datasets. As in other efforts in the search domain (e.g., schema.org) to specify a common vocabulary, one could leverage vocabularies to conceptualize relationships. Our DSKG includes nodes of different types, such as table, column, function, method, insight, and pattern. Some examples for edge types are: I) semantic similarity and inclusion dependency to interlink different data nodes, II) flows and reads to interlink code nodes together or to the used data nodes, and III) measure or aggregate to interlink insights related data nodes. We support automated or semi-automated maintenance of vocabularies to retain maximum flexibility.

Data Profiling: KEK data profiling aims at breaking down available artifacts into data items (columns, tables, datasets, pipelines, insights, etc.) to identify similarities and relationships. To achieve this goal, we will use the latest state of the art in data profiling and machine learning. KEK, for instance, requires the identification of hierarchies and statistics between data items such that this information can be used to construct a highly interconnected graph representation, in which vertices represent data items while edges represent relationships between them, such as "similarity". This graph is further annotated with provenance/metadata information and semantics to arbitrary domains of interest.

There is significant work in mapping columns and tables to concepts in knowledge graphs; but much of the work is primarily based on columns with string datatypes. More recent work has targeted numerical columns (e.g., [26]) but work of this nature is still at a fledgling stage. Our DSKGs are deductive graphs that utilize machine learning as well as inference rules to incrementally introduce and enhance the relationships among the different nodes in the graph. Therefore, the local DSKG will eventually be highly interconnected. This helps our profiling and construction process to scale to vast datasets.

Pipelines Abstraction: Similar programs are written with different APIs and languages. Initial efforts have been made to abstract the semantics of programs using static and dynamic program analysis techniques to extract language-independent representations of data science pipelines [2, 5]. Similar efforts capture the provenance of workflows, such as noWorkflow [33]. The example graph in Figure 1 (generated using [2]) illustrates how data flows through specific API pipeline calls, such as *SVM* or *SVC*. A key challenge that remains however is how one might recognize similar pipelines across frameworks or languages. There are many aligned benchmarks, such as CodeNet [34], that can be used by statistical models, such as Transcoder [37], to understand similarity across programs. One could leverage the associated natural language descriptions for APIs (e.g., documentation, forum posts) to generalize across multiple languages and frameworks. In Figure 1, for instance, the similarity of *SVC* and *SVM* could be derived from text, although this is still clearly an open challenge. Another challenge is to build multi-language independent abstractions for languages, that go beyond abstracting syntax trees. Systems, such as PROGRAML [7], derive abstract program graphs from neural models. These systems show initial promise for the development of language independent abstractions.

Insights Formulation: Data scientists use sophisticated libraries, such as R, Python, or Gnuplot, and tools, such as Tableau, Infogram, or Google Charts, for creating scripts capturing deeper insights from the data. While there are systems that have been proposed for extracting insights from an analysis of the data [9], they do not actually mine existing scripts targeting exploratory data analysis (EDA). Scripts targeting EDA are not easy to search; neither is it straightforward to enable automatic learning on them. There is a need for innovative approaches to capture the semantics of insights from the scripts, combined with comments in the scripts and connect them to their output including insights, observations, etc. Once this is accomplished, derived insights become searchable and processable at scale.

4.2 KEK Federated Services

The DSKG Construction analyzes the locally available datasets and scripts to build a local DSKG. The next step is to use the Federated Services to "connect" the local DSKG to the ones from other KEK portals via link prediction, as illustrated in Figure 2. We support federated querying, data enrichment, and pipeline automation on top of the decentralized DSKGs.

Link Prediction on DSKGs: In DSKGs, vertices represent data nodes, such as a node of type dataset, table, or column, or programming nodes, such as classes, functions, or methods, while edges represent relationships between these nodes, such as content similarity or function usage, respectively. We detect links between data items, such as tables or columns, using different methods, such as measuring content similarity. However, there are still other types of nodes or subgraphs, e.g., a pipeline or insights, where we need to predict links among them. We solve this problem as

a link prediction problem for knowledge graph completion using GNN-based models [50, 22]. KEK portals work transparently to interconnect different DSKGs and annotate DSKGs with provenance/metadata information. In KEK, learning the embeddings automatically is even more challenging due to the annotations in DSKG, i.e., hyper-relational facts [17], and the federated setup, which requires developing effective representation learning for datasets and data science artifacts in a geo-distributed environment.

Federated Querying and Exploration: Building upon knowledge graphs and existing standards, a variety of graph databases, commercial and research prototypes, is already available with basic support of federated querying. The challenge does not only lie within optimizing query execution across several KEK portals but also to keep each single one of them responsive despite potentially high query loads. Furthermore, KEK will support fine-grained and non-blocking query execution to produce results progressively. Thus, our federated execution model efficiently enables knowledge graph exploration and supports graph analytics queries generated by components, such as the semantic data enrichment and pipeline automation.

Semantic Data Enrichment: In the data preparation stage, data scientists tend to generate, in many cases, structured data, e.g., Dataframes, even from data sources of unstructured or semi-structured datasets, such as data logs or JSON documents. Usually, modeling results show data scientists that there is a need to add supplementary information to enrich the prepared dataset, as these dataframes may cover a limited number of cases. KEK assists users to easily extract relevant data, as discussed in Section 3.4. Moreover, KEK supports semantic data enrichment to find unionable, joinable, combinable data items, discover shortest paths, and schema integration. Users will be able to review discovered data before making the final decision on how to combine and further refine them. KEK further introduces functionalities to learn from the structure of DSKGs and make automatic recommendations for data enrichment based on semantic and syntactic matching.

Pipeline Automation Across Platforms: KEK's DSKG is able to capture API calls within a program, annotated with function calls and links to the used datasets. For pipelines, KEK does not join, i.e., combine two pipelines together. Instead, KEK interlinks similar pipelines to enable automatic graph learning for problems, such as pipeline automation as discussed in [20]. A DSKG takes the form of a knowledge graph and can be used in combination with deep graph generation networks [29] to model and generate pipelines for unseen datasets based on different representation learning techniques [47]. Then, we use state-of-the-art hyperparameter optimization systems, such as FLAML [43] or Auto-SKLearn [16], to recommend multiple optimized pipelines, see [20] for more details. Our model could be used by different ML platforms via KEK APIs to identify similar datasets to the unseen ones to generate new pipelines. Hence, KEK will provide a breakthrough for pipeline automation across platforms, i.e., by relying on the DSKGs, to help data scientists build data science pipelines quickly. There is a research opportunity to utilize the relevant datasets and previous analytical tasks to filter and classify generated pipelines.

4.3 DSKG Services

Graph Synchronization: KEK is not a static platform. As data scientists work on their projects and ideas, new datasets, pipelines, insights, etc., are continuously created. KEK platforms need to provide support to synchronize the local DSKG with local datasets and scripts of pipelines. This needs to incrementally maintain the DSKG and support pipelines generated by different platforms. This poses a research opportunity to develop a mechanism that efficiently updates the extracted semantics across scripts generated by different platforms.

Federated Graph Learning: KEK aims at developing a federated graph learning mechanism to learn graph representations (embeddings) across multiple DSKGs. KEK tasks, such as pipeline automation and semantic enrichment, benefit from this mechanism. We compute local and global features that generate embeddings based on the local and global DSKGs structure and topology. The graph features can be computed via analytical graph queries. Our federated graph learning is a promising technique to learn directly from the graph structure via sharing nodes' embedding with other remote connected nodes. This represents an open challenge for a scale message-passing framework in federated settings, and poses a research opportunity to develop an engine supporting variant embedding techniques for semantic queries [1]. This engine has to optimize the semantic query execution pipeline, automatically opt for the near-optimal embedding techniques, and estimate the cost of using this specific technique.

4.4 KEK Interface Services

For non-technical users, KEK provides question answering over DSKGs, automatically decide a data model for formalizing the results, and generate explanations.

Natural Language Questions: It is essential to reduce the technical effort required to explore and extract data/code from multiple KEK portals. Mapping a natural language question (NLQ) to a formal query language is challenging due to the ambiguity and multiple interpretations w.r.t. vertices related to data items, pipelines, and insights. Existing systems need thousands of annotated questions, such as NSQA [25], or require excessive preprocessing, such as such as gAnswer [21]. The preprocessing complexity is proportional to the KG size.

DSKGs are massive decentralized graphs that are fre-

quently updated. Thus, existing systems are impractical as the model should be re-trained from scratch for each update. There is a need for a model incrementally updated or trained independently of the graph. Thus, there is a need to develop a question answering system trained independently of the DSKG, as demonstrated by KGQAn [31]. The KGQAn system transforms a question into semantically equivalent SPARQL queries via a three-phase strategy based on natural language models trained generally for understanding and leveraging short English text. This poses a research opportunity to query multiple geo-distributed DSKGs and support natural language code and pipeline search [13].

Results Formulation and Explanation: Our methodology will develop different methods to estimate the query results' accuracy and index the NLQ segments and their relevant nodes and edges. The index will enhance the semantic understanding and linking of new NLQs based on the seen queries. The models will help in ranking query results. KEK's interface services should support data extraction in different formats based on the context of a given task and the NLQ semantic. For example, a data scientist may look for "Metro stations in Montreal," "Politicians born in New York City," or "Pipelines predicting car accidents in Aalborg". The result is not restricted to only one data model, e.g., a table format in the SQL language.

The result of these questions could be formalized as a map, table, or control flow graph, respectively. This represents an open challenge for adaptive models to predict the optimal formulation of results, e.g., as a table, graph, or map. Moreover, we need to annotate the results of NLQ with an explanation. Our methodology will adjust the query result's data model based on the NLQ semantics and its relevant data elements. This data model will include data explanations to help a data scientist understand the results in the context of a given task.

5. RELATED WORK

KEK is an end-to-end platform that enables the data science community to automatically discover, explore, and learn from existing data science artifacts and related datasets. The vision behind KEK is independent from or complementary to systems, such as Agora [39] or Cerebro [27], which focus on more technical aspects of executing data science pipelines across platforms, such as better utilization and unification of multiple computing resources or managing data as assets for trading, such as DMMS [15]. KEK, in contrast, is operating on a higher level of abstraction and could be built on top of the technical solutions provided by these systems.

In KEK, scripts of pipelines, and insights are managed by platforms of the user's choice. KEK captures the semantics of these scripts. Different tools, such as Vizier [4] and Ursprung [38], support the reproducibility of ML pipelines. The users can utilize these tools to manage their scripts without affecting KEK. LabBook [24] uses crowd sourcing to create a centralized knowledge graph to manage metadata about people, scripts and datasets, but KEK automatically extracts connections, in a highly distributed setting. AutoSuggest [48] is a tool helping in auto-completing a data-preparation pipeline. KEK focuses on modeling the detected insights and interlinking them with relevant datasets and pipelines. This will help automate several aspects of data science pipelines. Thus, these tools could benefit from KEK's knowledge graphs.

Systems, such as Google's Dataset Search Engine [30] and Helix [11], enable search over metadata of available datasets. Data discovery systems construct navigational data structures in the form of a linkage graph, such Aurum [14], an RDF knowledge graph, such as KGLac [19], or a hierarchical structure, such as RONIN [32]. Data sketches [28] can identify identical datasets used in different environments but cannot identify semantically similar data items or abstract a pipeline. Unlike these systems, KEK captures and extracts semantics of datasets, pipelines, and insights to construct a knowledge graph for data science enabling better collaboration in the community.

Multiple data versioning tools aim to track changes in the data used in ML models to enable reproducibility. Some tools were designed as S3 or Git extensions, such as Quilt [36], DVC [10], QRI [35], DataLad[8], and Git-LFS [18], to handle large data files. These tools do not handle schema changes, which may lead to breaking the execution of data science pipelines. Model management systems, such as ModelDB [42] and MLFlow [49], focus on reproducibility and tracing the modeling of experiments by capturing performance metrics, such as hyper-parameter and other values used in training. These data/model versioning tools do not capture the semantic abstraction of datasets and data science pipelines as proposed by KEK to enable advanced discovery and automatic learning.

6. CONCLUSION

KEK is a paradigm shift for open data science which brings together various communities, encourages more data scientists to share their work, and in doing so breaks down silos. In KEK, we utilize knowledge graph technologies to decouple the semantics of data science artifacts, e.g., pipelines and insights, from the data science platforms used to create and execute them. In doing so, KEK helps finding semantically similar artifacts and also finding out which artifacts should be combined to achieve a certain goal. The development of KEK poses numerous open research challenges that require innovative methodologies such as learning from decentralized knowledge graphs managed by geo-distributed KEK portals. In addition, new benchmarks are needed to mimic different workloads in federated data science.

7. REFERENCES

[1] H. Abdallah, D. Nguyen, K. Nguyen, and E. Mansour. Demonstration of KGNet: a cognitive knowledge graph platform. In *ISWC*, 2021.

[2] I. Abdelaziz, J. Dolby, J. P. McCusker, and K. Srinivas. A Toolkit for generating code knowledge graphs. *CoRR*, https://arxiv.org/abs/2002.09440, 2020.

[3] I. Abdelaziz, E. Mansour, M. Ouzzani, A. Aboulnaga, and P. Kalnis. Lusail: A system for querying linked data at scale. *PVLDB*, 11(4), 2017.

[4] M. Brachmann, W. Spoth, O. Kennedy, B. Glavic, H. Mueller, S. Castelo, C. Bautista, and J. Freire. Your notebook is not crumby enough, replace it. In *CIDR*, 2020.

[5] J. P. Cambronero and M. C. Rinard. AL: autogenerating supervised learning programs. *Proc. ACM Program. Lang.*, 3(OOPSLA):175:1–175:28, 2019.

[6] Canada Data Portal. https://open.canada.ca/.

[7] C. Cummins, Z. V. Fisches, T. Ben-Nun, T. Hoefler, and H. Leather. ProGraML: Graph-based deep learning for program optimization and analysis. *CoRR*, https://arxiv.org/abs/2003.10536, 2020.

[8] DataLad. http://www.datalad.org.

[9] R. Ding, S. Han, Y. Xu, H. Zhang, and D. Zhang. Quickinsights: Quick and automatic discovery of insights from multi-dimensional data. In *SIGMOD*, 2019.

[10] DVC. https://dvc.org.

[11] J. B. Ellis, A. Fokoue, O. Hassanzadeh, A. Kementsietsidis, K. Srinivas, and M. J. Ward. Exploring big data with Helix: Finding needles in a big haystack. *SIGMOD Rec.*, 43(4), 2014.

[12] Fei-Fei Li and Jia Li. Cloud AutoML: Making AI accessible to every business. *GOOGLE CLOUD*, 2018.

[13] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou. CodeBERT: A pre-trained model for programming and natural languages. In *EMNLP*, 2020.

[14] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. Aurum: A data discovery system. In *ICDE*, 2018.

[15] R. C. Fernandez, P. Subramaniam, and M. J. Franklin. Data market platforms: Trading data assets to solve data problems. *PVLDB*, 13(11):1933–1947, 2020.

[16] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter. Auto-Sklearn 2.0: Hands-free AutoML via meta-learning. *CoRR*, https://arxiv.org/abs/2007.04074, 2020.

[17] M. Galkin, P. Trivedi, G. Maheshwari, R. Usbeck, and J. Lehmann. Message Passing for Hyper-Relational Knowledge Graphs. In *EMNLP*, 2020.

[18] Git-lfs. https://git-lfs.github.com.

[19] A. Helal, M. Helali, K. Ammar, and E. Mansour. A demonstration of KGLac: A data discovery and enrichment platform for data science. volume 14, 2021.

[20] M. Helali, E. Mansour, I. Abdelaziz, J. Dolby, and K. Srinivas. A scalable AutoML approach based on graph neural networks. *CoRR*, https://arxiv.org/abs/2111.00083, 2021.

[21] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao. Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs. *TKDE*, 30(5):824–837, 2018.

[22] M. K. Islam, S. Aridhi, and M. Smail-Tabbone. A comparative study of similarity-based and GNN-based link prediction approaches. *CoRR*, https://arxiv.org/abs/2008.08879, 2020.

[23] Kaggle Portal. https://www.kaggle.com/.

[24] E. Kandogan, M. Roth, P. M. Schwarz, J. Hui, I. G. Terrizzano, C. Christodoulakis, and R. J. Miller. Labbook: Metadata-driven social collaborative data analysis. In *BigData*. IEEE, 2015.

[25] P. Kapanipathi, I. Abdelaziz, S. Ravishankar, and et al. Leveraging abstract meaning representation for knowledge base question answering. *CoRR*, https://arxiv.org/abs/2012.01707, 2020.

[26] U. Khurana and S. Galhotra. Semantic annotation for tabular data. *CoRR*, https://arxiv.org/abs/2012.08594, 2020.

[27] A. Kumar, S. Nakandala, Y. Zhang, S. Li, A. Gemawat, and KabirNagrecha. Cerebro: A layered data platform for scalable deep learning. *CIDR*, 2021.

[28] J. Lemiesz. On the algebra of data sketches. *PVLDB*, 14(9), 2021.

[29] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. W. Battaglia. Learning deep generative models of graphs. *CoRR*, http://arxiv.org/abs/1803.03324, 2018.

[30] N. Noy, M. Burgess, and D. Brickley. Google dataset search: Building a search engine for datasets in an open web ecosystem. In *WebConf*, 2019.

[31] R. Omar, I. Dhall, N. Sheikh, and E. Mansour. A Knowledge Graph Question-Answering Platform Trained Independently of the Graph. In *ISWC*, 2021.

[32] P. Ouellette, A. Sciortino, F. Nargesian, B. G. Bashardoost, E. Zhu, K. Pu, and R. J. Miller. RONIN: data lake exploration. *PVLDB*, 14(12), 2021.

[33] J. a. F. Pimentel, L. Murta, V. Braganholo, and J. Freire. NoWorkflow: A tool for collecting, analyzing, and managing provenance from python scripts. *PVLDB*, 10(12), 2017.

[34] R. Puri, D. S. Kung, G. Janssen, W. Zhang, G. Domeniconi, V. Zolotov, J. Dolby, J. Chen, M. R. Choudhury, L. Decker, V. Thost, L. Buratti, S. Pujar, and U. Finkler. Project CodeNet: A large-scale AI for code dataset for learning a diversity of coding tasks. *CoRR*, https://arxiv.org/abs/2105.12655, 2021.

[35] QRI. https://qri.io.

[36] Quilt. https://github.com/quiltdata/quilt.

[37] B. Rozière, M. Lachaux, L. Chanussot, and G. Lample. Unsupervised translation of programming languages. In *NeurIPS*, 2020.

[38] L. Rupprecht, J. C. Davis, C. Arnold, Y. Gur, and D. Bhagwat. Improving reproducibility of data science pipelines through transparent provenance capture. *PVLDB*, 13(12), 2020.

[39] J. Traub, Z. Kaoudi, J. Quiané-Ruiz, and V. Markl. Agora: Bringing together datasets, algorithms, models and more in a unified ecosystem [vision]. *SIGMOD Rec.*, 49(4), 2020.

[40] USA Data Portal. https://www.data.gov/.

[41] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations*, pages 49–60, 2014.

[42] M. Vartak and S. Madden. MODELDB: opportunities and challenges in managing machine learning models. *IEEE Data Eng. Bull.*, 41(4):16–25, 2018.

[43] C. Wang, Q. Wu, M. Weimer, and E. Zhu. FLAML: A fast and lightweight automl library. In *MLSys*, 2020.

[44] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific data*, 3, 2016.

[45] World Health Organization data portal. https://www.who.int/data/gho.

[46] World Trade Organization data portal. https://data.wto.org/.

[47] Z. Wu, S. Pan, F. Chen, and et al. A comprehensive survey on graph neural networks. *The IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2020.

[48] C. Yan and Y. He. Auto-Suggest: Learning-to-recommend data preparation steps using data science notebooks. In *SIGMOD*, 2020.

[49] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, F. Xie, and C. Zumar. Accelerating the machine learning lifecycle with mlflow. *The IEEE Data Engineering Bulletin*, 41(4):39–45, 2018.

[50] M. Zhang, P. Li, Y. Xia, K. Wang, and L. Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. *CoRR*, https://arxiv.org/abs/2010.16103, 2020.