Interpretable Feature Engineering for Structured Data

Mohamed Bouadi
Supervised by Arta Alavi and Prof. Salima Benbernou and Prof. Mourad Ouziri
SAP Labs Paris, Université Paris Cité, LIPADE
mohamed.bouadi@u-paris.fr

ABSTRACT

Machine Learning (ML) has demonstrated a significant utility in decision-making within the domain of data management. Since the quality of an ML model strongly depends on the quality of the input data, Feature Engineering (FE) stands as a pivotal step in enhancing the performance of these models, particularly with tabular data. However, traditional FE methods are often time-consuming and requires case-specific domain knowledge, which underscores the need for Automated Feature Engineering (AutoFE) techniques. In addition, with the proliferation of Data Analytics and ML-powered systems, especially in critical contexts, the need for interpretability and transparency becomes increasingly important, especially among domain experts. Studies have shown that ML models are only as interpretable as there features, highlighting the importance of feature interpretability in achieving ML interpretability. In this Ph.D. work, we tackle the problem of engineering interpretable features for structured data. We introduce *KRAFT*, an AutoFE framework harnessing a Knowledge Graph (KG) to guide the creation of interpretable features. Our approach integrates a neural generator for transforming raw features through diverse transformations with a knowledge-based reasoner for assessing feature interpretability. Additionally, we propose *ReaGen*, an AutoFE methodology combining KGs with large language models (LLMs) to produce interpretable features. This approach leverages external knowledge to extract relevant information related to the data and employs multiple LLMs to iteratively generate meaningful features based on dataset descriptions and retrieved information. Extensive experiments on real datasets validate the effectiveness of our solutions, showcasing notable improvements in accuracy and interpretability. Finally, we propose an interactive system dedicated to AutoFE techniques on structured data, allowing users to generate meaningful features and evaluate their efficiency and interpretability.

VLDB Workshop Reference Format:
Mohamed Bouadi. Interpretable Feature Engineering for Structured Data. VLDB 2024 Workshop: VLDB Ph.D. Workshop.

1 INTRODUCTION

Over the last decade, we have witnessed proliferation of ML applications on structured data to help solve difficult problems and uncover new opportunities across a variety of domains [1, 2]. The success of ML is often attributed to the experience of data scientists who leverage strong background in data mining and extensive domain knowledge to extract useful patterns from data. In most cases, appropriate transformation of data is an essential prerequisite step before model construction. This crucial task is commonly referred to as Feature Engineering (FE). Nevertheless, ML tools are still difficult to be utilized by non-experts, since a typical data science project contains many steps, as shown in Figure 1.

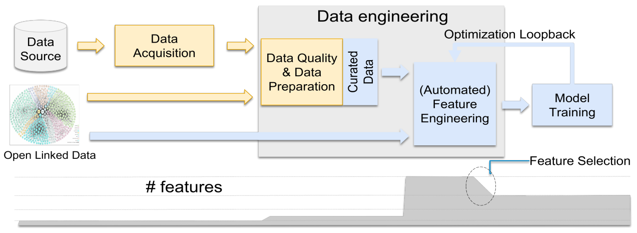


Figure 1: Data science workflow.

To reduce the workload of data scientists, automated ML (AutoML) has received an increasing interest. The reason is the resulting ability of organizations to automate the mundane and create business solutions to solve real world problems. However, FE remains a significant bottleneck in the data science workflow due to the combinatorial nature of the search space. Initially, this space is given by the following equation:

$$Dom_F = \bigcup_{i=1}^{p} \left\{ \left\{ \bigcup_{1 \leq s_1 \leq .. \leq s_i \leq p} \{(f_{s_1}, .., f_{s_i})\} \right\} \times t_i \right\}, \quad (1)$$

where p is the number of features and $t_i \subseteq \mathcal{T}$ is the set of i-ary functions. The number of elements in this space is: $|Dom_F| = \sum_{i=1}^{p} (A_i^p \times |t_i|)$, where A_i^p is the i-permutation of p features.

This space grows exponentially even with a limited number of transformations, hence an exhaustive search is not feasible. In addition, FE necessitates extensive domain knowledge. Transforming raw data into meaningful features often requires deep insights into the domain from which the data originates. It is even more challenging with diverse data sources requiring to wisely identify and execute relevant joins and aggregates to select the best features. This tedious task requires up to 80% of the data scientist's total time, as reported in the State of Data Science Report[1]. Moreover, feature interpretability remains an open challenge in AI. ML models are only as interpretable as their features [14]. However, existing AutoFE approaches often struggle to generate interpretable features.

Consequently, automating FE is becoming increasingly attractive as it may reduce data scientists workload significantly so they can make quick decisions with lower costs. This resulted in various approaches. The first one generates a large number of features

Proceedings of the VLDB Endowment. ISSN 2150-8097.

[1]https://www.anaconda.com/state-of-data-science-report-2023

followed by pruning and selection [10, 13]. However, these techniques suffer from performance bottleneck due to the large number of candidates. Another approach evaluates the usefulness of each feature through training and evaluation [7, 9]. While conceptually promising, these approaches are slower due to extensive model training and fail to address the problem of feature interpretability.

In order to address the aforementioned challenges, we propose the following research directions:

- We study features properties and propose a metric to evaluate feature interpretability using a Description Logics-based reasoner (DL) over a KG, that captures domain knowledge.
- To addresses the limitations of traditional feature-based representations in neural networks, which lack semantics and domain knowledge, we propose a new semantic vectorization technique to represent the input features with a feature vector based on the semantics embedded in KG [6].
- We propose *KRAFT*, a knowledge-driven solution that takes advantage of the semantics embedded in the domain knowledge to automatically engineer interpretable features using a Deep Reinforcement Learning agent (DRL).
- We combine the capabilities of LLMs with knowledge-based reasoning techniques to automate the process of FE.

2 BACKGROUND

2.1 Feature Interpretability

ML models need to learn from good data. Even simple and interpretable models, like regression, become difficult to understand with non-interpretable features. However, there is no consensus regarding the definition of interpretability in ML and how to evaluate it. It usually refers to the "ability to present in understandable terms to a human". In our research, we are focusing on *feature interpretability* for domain experts by capturing the connections between features semantics and domain knowledge. To this end, we propose the following definition:

Definition 2.1. (Feature Interpretability). We define feature interpretability as the ability of domain experts to understand a feature and relating it to relevant concepts within their domain knowledge. This implies mapping every new feature to relevant intensional and extensional knowledge within the domain of interest.

Consequently, interpretable features should be humanly-readable and refer to real-world entities that domain experts can understand and reason about.

2.2 Feature Engineering

Consider a predictive problem on a tabular dataset $D = (X, Y)$ consisting of: (i) a set of features $X = \{x_1, .., x_p\} \in \mathbb{R}^{n \times p}$, where n is the number of instances and p is the number of features; (ii) an applicable ML algorithm, L, (e.g. Random Forest); (iii) a corresponding cross-validation performance measure $\mathcal{P}$ (e.g. F1-score); and (iv) an interpretability function, $I_{KG} : Dom^F \longrightarrow \{0, 1\}$, where Dom^F is the set of all possible features, that returns 1 if the feature is interpretable and 0 otherwise. We define a FE pipeline $\mathcal{T} = \{t_1, ..., t_m\}$ as an ordered sequence of m transformations applied to X. The set of generated features from X using $\mathcal{T}$ is denoted as $\hat{X}_{\mathcal{T}}$.

The goal of AutoFE is to find the optimal FE pipeline, $\mathcal{T}$, that generates $\hat{X}_{\mathcal{T}}$ which maximizes the performance $\mathcal{P}(L(\hat{X}_{\mathcal{T}}, Y))$ for a given algorithm L and a metric $\mathcal{P}$, as shown in Equation 2:

$$\mathcal{T} = \arg\max_{\mathcal{T}} \mathcal{P}(L(\hat{X}_{\mathcal{T}}, Y)) \quad s.t. \quad \prod_{\hat{x_i} \in \hat{X}_{\mathcal{T}}} \mathcal{I}_{KG}(\hat{x_i}) = 1, \mathcal{I}_{KG}(\hat{x_i}) \in \{0, 1\}, \forall \hat{x_i} \in \hat{X}_{\mathcal{T}} \tag{2}$$

3 PROPOSED SOLUTIONS

In this section, we present our proposed solutions. The first one, *KRAFT*[2] [4, 5], leverages a KG to guide the generation of interpretable features through DRL. The second, *ReaGen*[3], combines LLMs with KGs to produce interpretable features. In addition, we recently proposed an interactive system that allows users to generate meaningful features, using different AutoFE approaches, and evaluate their efficiency and interpretability.

3.1 KRAFT

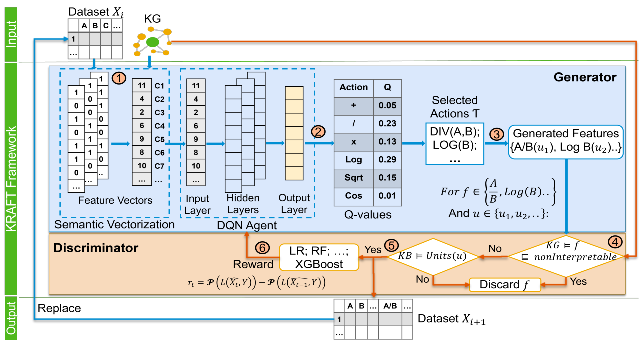


Figure 2: An overview of *KRAFT* architecture.

Kraft consists of two parts: a *Generator* and a *Knowledge-based Discriminator*, as illustrated in Figure 2.

3.1.1 Generator. We consider FE on a dataset D as a Markovian Decision Process [3]. At each step t, a state $s_t \in S$ represents the current dataset. The set of actions, A, corresponds to the set of transformations $\mathcal{T}$. The reward is the average performance gained from the previous step on a k-fold cross validation, i.e., $r_i = \mathcal{P}(L(\hat{X}_i, Y)) - \mathcal{P}(L(\hat{X}_{i-1}, Y))$.

The generator is modeled by a policy network $\pi : S \longrightarrow Q(A)$, where Q is the expected cumulative reward estimated by the agent. The goal is to obtain the best sequence of m transformations. We used 2 instances of a fully-connected multi-layer neural network to approximate the Q-function: (i) The main neural network $Q(s, a; \theta_i)$ to estimate the cumulative reward for each possible action; (ii) The target network, $\hat{Q}(s', a'; \hat{\theta}_i)$ with delayed updates. A Q-learning

[2] Leveraging Knowlegde Graphs for Interpretable Feature Generation

[3] Synergizing Large Language Models and Knowledge-based Reasoning for interpretbale Feature Generation

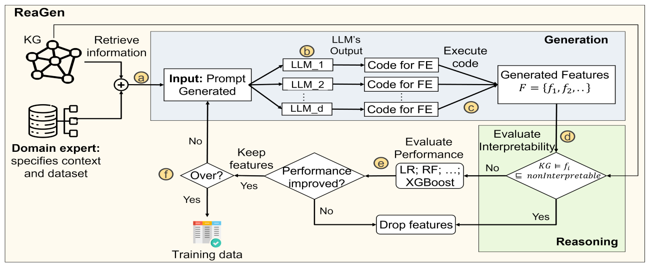


Figure 3: An overview of *ReaGen* architecture.

update at iteration i is defined as the loss function:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} \hat{Q}(s', a'; \hat{\theta}_i) - Q(s, a; \theta_i)\right)^2\right] \tag{3}$$

Basically, at each step of the training, and iteratively until convergence, the agent receives the current state and feed it to the neural network. This latter calculates an intermediate reward for each possible action and selects the one that maximizes the long term reward and this action is used to generate new features.

3.1.2 Knowledge-based Discriminator. The discriminator is a reasoning algorithm used to discard non-interpretable features. We used HermiT [12], a reasoner for the DL syntax, to infer new knowledge based on the logical relationships of the KG. To decide if a feature $\hat{x} \in \hat{X}_{\mathcal{T}}$ with unit u is interpretable, the discriminator considers $\hat{x}$ as a DL concept and performs a *subsumption reasoning*. First, it checks if $\hat{x}$ can be subsumed from the class *non-interpretable*, i.e., $KB \models \hat{x} \sqsubseteq$ *non-interpretable* (Figure 2.4). In this case, $\hat{x}$ would be removed. However, if we do not have enough information about the feature, the discriminator uses the knowledge about its unit. To this end, it performs *instance checking* ($KB \models Units(u)$), as shown in Figure 2.5. If the unit is unknown, $\hat{x}$ would be considered non-interpretable. In this way, the discriminator discards all non-interpretable features and ensures that the selected features and transformations are understandable to domain experts.

3.2 ReaGen

As shown in Figure 3, *ReaGen* takes a dataset along with its description as inputs. Initially, it leverages external knowledge (e.g., KG, Ontologies) to extract additional information and entities (Figure 3.a). Based on a predefined template, *ReaGen* generates a prompt that is used to iteratively generate new features (Figure 3.b). Each iteration begins with the generation of new features using multiple LLMs as illustrated in Figure 3.c. The LLMs also provide explanations that justify the utility of each feature. These explanations are crucial for assessing the interpretability and relevance of the generated features. To filter out non-interpretable features, *ReaGen* employs the discriminator from our previous solution, *KRAFT*, that ensures the generated features and transformations used are understandable to users (Figure 3.d). Once interpretable features are obtained, they are evaluated using cross-validation metrics. If the performance metrics improve compared to the previous iteration, the new features are retained as shown in Figure 3.e.. This iterative approach allows *ReaGen* to refine the generated features and enhance the performance of the augmented dataset.

3.3 DANTE

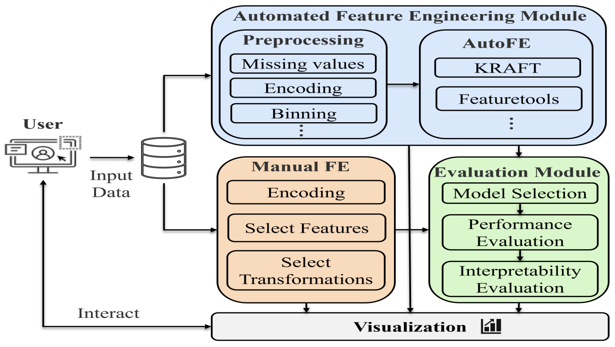
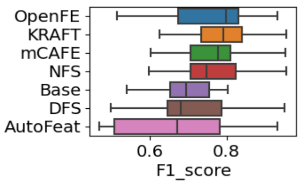


Figure 4: An overview of DANTE architecture.

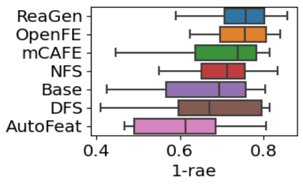
DANTE, shown in Figure 4, comprises four main modules: (1) AutoFE, (2) Manual FE, (3) Evaluation, and (4) the Graphical User Interface (GUI). **AutoFE module** enables users to automatically handle missing values and categorical features. In addition, it facilitates automatic generation of interpretable features via our approach *KRAFT*. It also integrates other AutoFE techniques, like FeatureTools [10], in order to offer flexibility and alternative FE strategies to the user. **Manual FE module** offers an interactive FE environment where users manually select the transformations to be applied on features to generate new ones. **The Evaluation module** adds a performance assessment layer to the system. It lets the users select from a list of SOTA ML models and cross-validation metrics for model evaluation. Users can explore the results through different types of graphs and tables. Finally, **the GUI** provides a user-friendly platform for interacting with the system. Users can upload their own datasets or select from available ones, choose an ML model and performance metric, and select a FE method.

4 EMPIRICAL EVALUATION

We now present some results of our experimental evaluation of *KRAFT* and *ReaGen*. We used several public datasets to compare our methods with SOTA approaches of AutoFE. Due to space constraints, Figure 5a presents the *F1-score* comparison between our



(a) *KRAFT* versus baselines.



(b) *ReaGen* versus baselines.

Figure 5: Overview of evaluation results: (a) Avg. F1_Score of KRAFT; (b) Avg. RAE of *ReaGen* across all datasets.

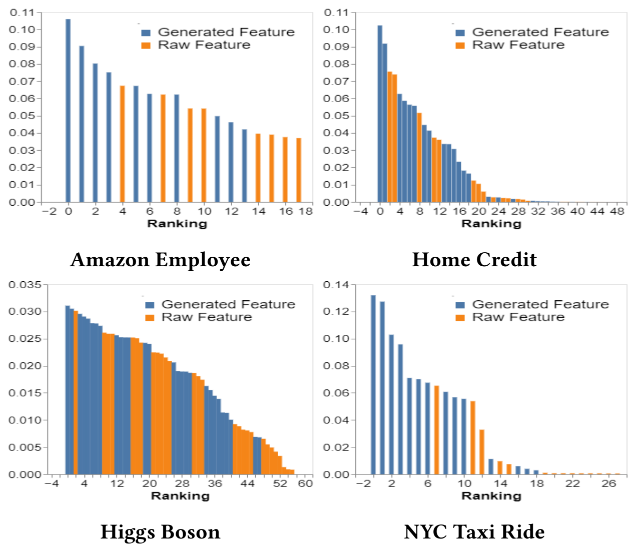


Figure 6: Feature importance of raw features (in orange) and generated features with *KRAFT* (in blue).

KRAFT and SOTA approaches for classification tasks, while Figure 5b illustrates the *Relative Absolute Error (RAE)* comparison of *ReaGen* with SOTA approacges for regression tasks. Our experiments showed that both *KRAFT* and *ReaGen* outperformed the SOTA AutoFE across all datasets.

Additionally, in Figure 6, we show the contribution of *KRAFT*-generated features (blue) compared to raw features (orange) to the final prediction across 4 datasets, using SHAP (SHapley Additive Explanations) [11], a model-agnostic interpretability approach. We created a new dataset by combining the *n* raw features with the top-ranked *n* features generated by *KRAFT* and used SHAP to score the feature contributions. Notably, *KRAFT*-generated features are consistently more important than raw features across all datasets, offering insights into the model's global behavior, making it more transparent and trustworthy for users.

5 ONGOING WORK AND CONCLUSION

In this paper, we introduce our solutions to tackle the problem of engineering interpretable features for structured data. The first solution, *KRAFT* [4, 5], is based on two components: a DRL agent to generate new features and a human-like reasoner that uses a KG to filter out non-interpretable features. The second one, *ReaGen*, combines the use of LLMs with a knowledge-based reasoner to automate FE. By incorporating external knowledge, *ReaGen* enhances the factual accuracy and grounding on LLMs, thereby reducing the occurrence of hallucinated thoughts. Finally, we present *DANTE*, an interactive system that demonstrates the practical utility of *KRAFT* allowing users to generate new features as well as to evaluate their efficiency and interpretability without any code. Extensive experiments on large scale datasets are conducted, which show that our methods can provide competitive performance and guarantee the interpretability of the generated features.

In this Ph.D. work, we aim to bridge the gap between ML and symbolic AI, highlighting the importance of incorporating domain knowledge and context-aware solutions in AutoML tools. The ongoing work and future research directions include the following:

(1) *KRAFT* uses a binary interpretability function using DL. In future work, we aspire to investigate the use of other symbolic AI approaches to go towards a flexible graduated framework of interpretability evaluation, such as Probabilistic Logics and Fuzzy sets [8].
(2) Design more customized metrics to evaluate different aspects of feature interpretability, taking into consideration the profile of users.
(3) Develop more customized architectures for FE. An interesting candidate would be to replace the DRL agent with transformers architecture. These models have shown impressive results in encoding contextual information and capturing dependencies between different parts of a sequence.
(4) Build a foundation model specialized in context-aware feature generation for tabular data, and foresee the potential of LLMs to automate other steps in the data science workflow, such as model selection for tabular data.

REFERENCES

[1] Sihem Amer-Yahia, Reynold Cheng, Mohamed Bouadi, Abdelouahab Chibah, Mohammadreza Esfandiari, Jiangping Zhou, Nan Zhang, Eric Lau, Yuguo Li, Xiaolin Han, et al. 2020. An ML-Powered Human Behavior Management System. *Bulletin of the Technical Committee on Data Engineering* 43, 3 (2020), 53–64.
[2] Idir Benouaret, Mohamed Bouadi, and Sihem Amer-Yahia. 2021. Multi-Objective Recommendations and Promotions at TOTAL. In *Database and Expert Systems Applications: 32nd International Conference, DEXA 2021, Virtual Event, September 27–30, 2021, Proceedings, Part II 32*. Springer, 270–282.
[3] Mohamed Bouadi and Arta Alavi. 2023. Automated feature engineering for predictive modeling using deep reinforcement learning. US Patent App. 17/694,288.
[4] Mohamed Bouadi, Arta Alavi, Salima Benbernou, and Mourad Ouziri. 2024. Leveraging Knowlegde Graphs for Interpretable Feature Generation. *arXiv preprint arXiv:2406.00544* (2024).
[5] Mohamed Bouadi, Arta Alavi, Salima Benbernou, and Mourad Ouziri. 2024. Ontology-based framework for interpretable feature engineering. US Patent App. 18/178,768.
[6] Mohamed Bouadi, Arta Alavi, Salima Benbernou, and Mourad Ouziri. 2024. Semantic vectorization for feature engineering. US Patent App. 18/168,982.
[7] Xiangning Chen, Qingwei Lin, Chuan Luo, Xudong Li, Hongyu Zhang, Yong Xu, Yingnong Dang, Kaixin Sui, Xu Zhang, Bo Qiao, et al. 2019. Neural feature search: A neural architecture for automated feature engineering. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 71–80.
[8] Víctor Gutiérrez-Basulto, Jean Christoph Jung, Carsten Lutz, and Lutz Schröder. 2017. Probabilistic description logics for subjective uncertainty. *Journal of Artificial Intelligence Research* 58 (2017), 1–66.
[9] Yiran Huang, Yexu Zhou, Michael Hefenbrock, Till Riedel, Likun Fang, and Michael Beigl. 2022. Automatic Feature Engineering Through Monte Carlo Tree Search. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 581–598.
[10] James Max Kanter and Kalyan Veeramachaneni. 2015. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 1–10.
[11] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).
[12] Robert DC Shearer, Boris Motik, and Ian Horrocks. 2008. Hermit: A highly-efficient OWL reasoner.. In *Owled*, Vol. 432. 91.
[13] Tianping Zhang, Zheyu Aqa Zhang, Zhiyuan Fan, Haoyan Luo, Fengyuan Liu, Qian Liu, Wei Cao, and Li Jian. 2023. Openfe: Automated feature generation with expert-level performance. In *International Conference on Machine Learning*. PMLR, 41880–41901.
[14] Alexandra Zytek, Ignacio Arnaldo, Dongyu Liu, Laure Berti-Equille, and Kalyan Veeramachaneni. 2022. The need for interpretable features: motivation and taxonomy. *ACM SIGKDD Explorations Newsletter* 24, 1 (2022), 1–13.