Semantic Feature Discovery with Code Mining and Semantic Type Detection

Kavitha Srinivas, Takaaki Tateishi, Daniel Karl I. Weidele, Udayan Khurana*, Horst Samulowitz, Toshihiro Takahashi, Dakuo Wang, Lisa Amini

IBM Research

Abstract

In recent years, the automation of machine learning and data science (AutoML) has attracted significant attention. One under-explored dimension of AutoML is being able to automatically utilize domain knowledge (such as semantic concepts and relationships) located in historical code or literature from the problem's domain. In this paper, we demonstrate *Semantic Feature Discovery*, which enables users to interactively explore features semantically discovered from existing data science code and external knowledge. It does so by detecting semantic concepts for a given dataset, and then using these concepts to determine relevant feature engineering operations from historical code and knowledge.

Introduction

Automated Machine Learning (AutoML) refers to the techniques of automatically building predictive models. It has gained popularity in recent years as it promises to save time and reduce the cost of data science projects (Aggarwal et al. 2019). A critical step of data science pipelines is *feature engineering* – the process of expanding and augmenting the feature space of the given data with a goal to optimize model performance. Researchers have also proposed various optimization methods to automate the feature engineering process (Khurana et al. 2016). However, so far automation is not able to identify semantic meanings from the data or applying domain knowledge, as a human data scientist would do. They are based solely on statistical or brute-force approaches. Additionally, the features generated by these automated systems typically lack semantically relevant explainability, thus users find it hard to interpret.

In this paper, we demonstrate a novel system, *Semantic Feature Discovery (SFD)*, which can automatically discover concepts in the given data, then links these concepts to external knowledge and existing code repositories. With its novel algorithm design, SFD can generate not only semantically rich features to improve the accuracy of the predictive models, but also human-readable explanations with these features. SFD is aligned with IBM's AutoAI[1], demonstrating an end-to-end AutoML capability with the human in the loop. In this paper, we highlight the novel algorithmic and system design contributions of SFD, in comparison to prior semantic FD techniques (Galhotra et al. 2019).

System Overview

In order to add the novel dimension of semantics to assisting Data Scientists we present a two-phased approach. The first aims at mapping concepts to features of given data. Once data is annotated - even if it is done manually for some features - it enables automated discovery of relationships not only between existing features, but also to external or not yet considered data. While concept and relationship discovery is very valuable for data understanding and a goal of this work as well, it cannot be leveraged directly for automatically generating new features. Consider the example where we mapped one feature to 'total confirmed COVID cases' and another to 'Country/Region'. Now, one common feature to add would be 'percentage of confirmed cases per region'. In this example, trial and error of semantically uninformed optimization might have discovered this relationship, but in our second approach we leverage existing Data Science code to extract and apply suitable data engineering steps based on semantic understanding. To do so, the concepts derived with the first approach are leveraged in the second to identify which code fragments are relevant to the data at hand.

Column to Concept Mapping

Determining concepts represented by columns in a given ML problem is the essential first step in performing semantic feature engineering. We use two column to concept mappers to perform this task. The first called C^2 (Khurana and Galhotra 2021) is based on a Maximum Likelihood approach through ensembles which computes MLL based on actual cell values of a column. It maximizes the likelihood of a concept across different sources of structured data (data table lakes such as Manyeyes, Opendata, Wikipedia tables; knowledge graphs such as Wikidata and DBpedia), and acrosss various cells of a column.

The second tries to map column names to concepts in knowledge graphs based on the following three functions: (1) a notation fluctuation solver which uses word2vec-like phrase similarity metrics to identify different phrases in column names (e.g., "speed" and "velocity") mapped to the

*Corresponding author: ukhurana@us.ibm.com

[1]IBM Watson AutoAI: https://www.ibm.com/in-en/cloud/watson-studio/autoai

same concept, (2) an alias solver which uses alias information explicitly defined in Wikidata or inferred based on the anchor texts of links in Wikipedia, to understand different word level references to the same concept, (3) a context understanding component which uses TF-IDF based sentence similarity metrics between the column descriptions and sentences of Wikidata & DBpedia to distinguish concepts that have the same label (e.g., apple of Apple Inc. or fruit).

Code Mining

With the availability of large repositories of code, it is possible to mine code for feature engineering that involves the use of the same columns. We use three components to mine code for feature engineering: (a) We have developed a tool to perform to perform interprocedural static analysis for Python code for millions of Python programs (Srinivas et al. 2020)[2], which we use to identify reads and writes to objects such as `pandas.DataFrame` to identify columns. We then extract and generalize expressions from analysis of the form `A = B-C` from mining code. (b) We have developed a tool that automatically instruments code (Gosain and Sharma 2014) for dynamic analysis, such that the tool executes data science programs to keep track of execution traces, including method calls on objects such as `pandas.Series` and their arguments, from which we extract expressions (e.g., `A=B-C`) over column names. The instrumention code handles mock objects and ignores runtime errors caused by the method calls such as `KeyError` so that we can run the instrumented program without errors even if no actual dataset is given. (c) We mine text cells in Jupyter notebooks for formula expressions in Latex and use question answering techniques such as RoBERTa fine tuned on SQUAD (Liu et al. 2019) to understand the terms of each formula. So a formula expressed in text as `A = B-C`, gets converted into a series of questions such as "What is A?" to extract the meaning for each term.

When possible, the extracted expressions and formulas are then generalized by mapping the column names to corresponding concepts from the column to concept mapping component above.

SFD Demonstration

AutoAI is a tool to assist Data Scientists with model building task such as model selection, parameter search and 'statistical' feature engineering. As in most such tools its starting point requires the user to provide data. In this demonstration we choose a publically available dataset on Covid [3] with the the following features: $Date$, $Longitude$, $Latitude$, $Province/State$, $Country/Region$, $Recovered$, $Confirmed$, $Deaths$.

Before kicking off the automated model-building, interactive SFD is invoked. In the UI, the provided data is first automatically annotated with broad domain concepts. Note that this example data does not explicitly mention 'Covid', but SFD's concept mapper is able to infer it. Figure 1 captures how SFD automatically annotates the orginal columns (blue dots) with concepts (green tags) which can be further

[2]https://wala.github.io/graph4code/

[3]https://www.kaggle.com/gpreda/coronavirus-2019ncov

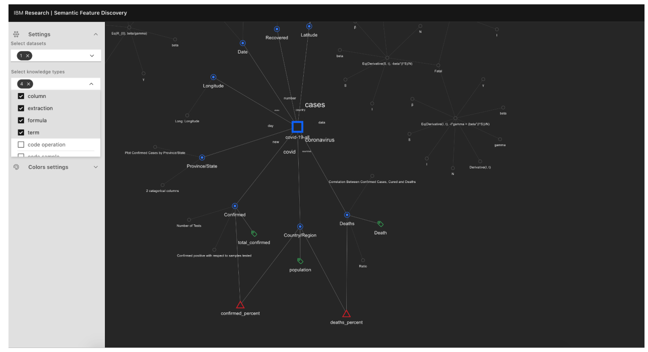

Figure 1: Browsing concept and expression relationships for a column

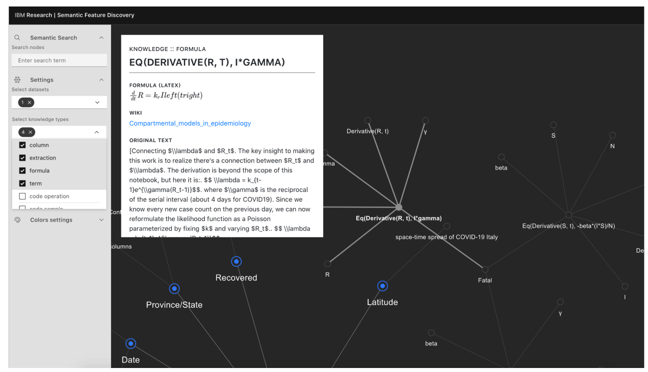

Figure 2: Browsing formulas relevant for a domain

refined by the user. For instance, *Country/Region* is tagged with *Population*. Using available concepts, SFD maps directly computable relationships (red triangles) to the corresponding concepts that the user can choose to apply after inspecting the code and context it was mapped from. When applied, the code for features is generated and executed as a data transformation step in the data science workflow.

SFD does not only discover directly applicable relationships, but also relationships that are not associated with a computable action. For example, SFD suggests to consider the epidemology model to compute the spread of a disease (Figure 2). While SFD is not able to extract enough information to automatically employ this relationship, it certainly provides the Data Scientist with insights and assistance that goes far beyond what current tools accomplish. Note that the syntactial errors of the text box in Figure 2 are direct results of automated retrieval and underlining the fact that more processing is needed.

Once the Data Scientist has concluded the data discovery and augmentation stage, the desired data transformation are applied and model building is initiated. While common sense reasoning and understanding domain knowledge remain mostly elusive tasks in modern AI, we demonstrate that some limited use of semantics combined with Human-Computer interaction can add a new and powerful dimension to assisting Data Scientists.

References

Aggarwal, C.; Bouneffouf, D.; Samulowitz, H.; Buesser, B.; Hoang, T.; Khurana, U.; Liu, S.; Pedapati, T.; Ram, P.; Rawat, A.; et al. 2019. How can ai automate end-to-end data science? *arXiv preprint arXiv:1910.14436*.

Galhotra, S.; Khurana, U.; Hassanzadeh, O.; Srinivas, K.; Samulowitz, H.; and Qi, M. 2019. Automated Feature Enhancement for Predictive Modeling using External Knowledge. In *IEEE ICDM*.

Gosain, A.; and Sharma, G. 2014. A Survey of Dynamic Program Analysis Techniques and Tools. In Satapathy, S. C.; Biswal, B. N.; Udgata, S. K.; and Mandal, J. K., eds., *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)*, volume 327 of *Advances in Intelligent Systems and Computing*, 113–122. Springer.

Khurana, U.; and Galhotra, S. 2021. Semantic Annotation for Tabular Data. *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*.

Khurana, U.; Turaga, D.; Samulowitz, H.; and Parthasrathy, S. 2016. Cognito: Automated Feature Engineering for Supervised Learning. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 1304–1307. IEEE Computer Society.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, abs/1907.11692.

Srinivas, K.; Abdelaziz, I.; Dolby, J.; and McCusker, J. P. 2020. Graph4Code: A Machine Interpretable Knowledge Graph for Code. *CoRR*, abs/2002.09440.