LDK 2023

Pruning and re-ranking the frequent patterns in knowledge graph profiling using machine learning

Gollam Rabby
L3S, Leibniz University Hannover, Hanover, Germany
and
VSE, Prague, Czechia
gollam.rabby@L3S.de

Farhana Keya
TIB Leibniz ICST, Hanover, Germany
and
VSE, Prague, Czechia
keya@tib.eu

Vojtěch Svátek
VSE, Prague, Czechia
svatek@vse.cz

Blerina Spahiu
University of Milano-Bicocca, Milan, Italy
blerina.spahiu@unimib.it

Abstract

Sets of frequent schema-level patterns characterizing a given knowledge graph (KG) represent a central output of profiling tools such as ABSTAT, as they could provide a quick overview of the coverage of the KG and its adequacy for various tasks. However, the number of patterns may be huge. The most frequent ones are often not useful for semantically characterizing the KG since they feature generic (OWL, SKOS, etc.) classes and even XML data types. We hypothesize that the pattern profile suitability for a 'rapid skimming' scenario might be improved by applying pattern post-processing, namely, their pruning and/or re-ranking. In this paper, we investigate, for this purpose, different machine learning (ML) methods trained on manually labelled examples (whole namespaces or individual IRIs of entities). Random Forest, Decision Tree and Multi-layer Perceptron Classifiers get higher accuracy than others.

1 Introduction

Because of the high number and large size of knowledge graphs (KGs), which makes it difficult to rapidly identify the KG suitable for a particular application, KG *profiling* was recently introduced as a means of quantifying the structure and contents of KGs to judge their suitability for particular applications. Of the many quantitative and qualitative characteristics that can describe a KG, the schema-level pattern of the form `<subjectType, pred, objectType>` as an abstract representation of the KG instances is particularly interesting from the point of view of knowledge engineering. Profiling tools based on schema patterns, such as ABSTAT (Spahiu et al., 2016) or Loupe (Mihindukulasooriya et al., 2015), give the user specific insights into frequent paths interconnecting entities at the instance level while remaining relatively concise. The outcome depends on the ontology employed and the degree of explicit typing of entities. The internals of these tools consist of sophisticated graph-theoretic methods, and some rely on massive parallelization of the computation. However, the results in their generic form may not always fit every kind of usage. The scenario we have in mind is that of *rapid skimming through multiple KGs* to identify those having adequate coverage of some topic/s (contrasting to a scenario requiring detailed scrutiny of a dataset's schema). For this, the output of a state-of-the-art tool such as ABSTAT (even a 'minimal,' non-redundant set) still contains too many patterns that are 'boring' concerning such skimming.

In our previous work (Rabby et al., 2022) we directly applied a handful of manually-written heuristics in order to (further) prune as well as re-rank the output of ABSTAT. The current paper extends this previous attempts by exploring, for the same purpose, various machine learning (ML) methods which have been trained on manually labeled examples.

2 ABSTAT

ABSTAT is a scalable profiling tool that aims to support users in exploring and understanding large RDF KGs. Given a KG in the form of a dataset and an ontology (optional), ABSTAT computes a profile comprising a summary of the dataset content and statistics. A summary is a set of data-driven ontology patterns in the form `<subjectType, pred, objectType>`, which represent the occurrence of the triples `<subj, pred, obj>` in the dataset. Minimalization is applied on types and properties; that is, `subjectType` is a minimal type for `subj` (i.e., no type for `subj` is in subsumption relation with `subjectType`), `objectType` is a minimal type of the `obj` and `subj` is linked to `obj` through `pred` or any other super-property of `pred`, at this moment defining a clear distinction between patterns (a redundant pat-

Table 1: The distribution of the categories with frequency.

Category	Frequency
Remove	216
Put to the bottom	179
None	306

Table 2: Accuracy, Macro, and Weighted average for the different machine learning methods; RF = Random Forest; LinearSVC = Linear Support Vector Classifier; LR = Logistic Regression; MultinomialNB = Multinomial Naive Bayes; KNeighbors = K-Nearest Neighbors; SVC = Support Vector Classifier; DT = Decision Tree; MLP = Multi-layer Perceptron Classifier; AdaBoost = Adaptive Boosting Classifier.

ML methods	Accuracy	Macro avg	Weighted avg
RF	**0.49**	0.45	0.49
LinearSVC	0.48	0.45	0.45
LR	0.48	0.45	0.45
MultinomialNB	0.48	0.44	0.44
KNeighbors	0.43	0.38	0.38
DT	**0.49**	0.46	0.46
MLP	**0.49**	0.46	0.45

tern set) and minimal patterns. We will henceforth refer to minimal patterns as patterns. In addition, statistics such as the frequency of how many assertions in the dataset are represented by each pattern are also extracted. (Spahiu et al., 2016) describes the details of this KG profiling tool. The pruning effect of minimization becomes more effective when at the same time, ontologies encode a rich type hierarchy, and entities are primarily associated with many types (e.g., DBpedia). However, since ABSTAT is designed to summarize assertions in the KG while maintaining full coverage of them, it could be that a KG featuring many entities without a type and with a poor (absent) type hierarchy, fed to ABSTAT, leads to a summary with some pattern which may not be informative to the user because of its high generality.

3 Methods

The motivation for post-processing is to suppress the patterns that contain overly general namespaces or individual schema IRIs, so that, ideally, only patterns expressing ontological relationships properly characterizing the KG are left (thus also reducing the overall size of the pattern set) or at least prioritized in the list.

Input data To create the input dataset for manual labelling, we generated a list of frequent KGs patterns produced by ABSTAT (as stored in its database), and collected the IRIs of all entities appearing in them. This became a basis for a table to be used by human annotators, which contained 700 randomly picked entities. Three annotators (from among the paper authors) eventually labelled about 400-500 of them each, using a set of three labels: "None", "Put to the bottom", and "Remove". A single label for each IRI was obtained by majority vote. The frequency count of the ultimate values is in Table 1.

Entity representation The Term Frequency and Inverse Document Frequency (TF-IDF) is one of the most popular text representation methods, widely employed in numerous previous studies. To construct the TF-IDF input data table, our experiment used the unigrams and bigrams extracted from the (parsed) entity IRI.

Machine learning methods We used the random forest (Breiman, 2001), linear support vector classifier (Suthaharan and Suthaharan, 2016), logistic regression (LaValley, 2008), multinomial naive bayes (Xu et al., 2017), K-Nearest neighbors (Peterson, 2009), decision tree (Safavian and Landgrebe, 1991) and multi-layer perceptron classifier (Ramchoun et al., 2016) implementation from the scikit-learn library, with hyperparameter optimization (see Table 3). We also utilized the k-fold cross-validation from the scikit-learn. It provides cross-validation with grid search hyperparameter

Table 3: Overview of input Parameter grid (Optimal configurations are bold).

Machine learning algorithm	Parameter grid
Random Forest	'n_estimators': [100, **200**, 300], 'max_depth': [**2**, 5, 10], 'min_samples_split': [2, 5, **10**], 'min_samples_leaf': [1, **2**, 4]
Linear Support Vector Machine	'C': [**0.1**, 1, 10], 'loss': ['hinge', '**squared_hinge**'], 'max_iter': [**1000**, 2000, 3000]
Logistic Regression	'C': [0.1, **1**, 10], 'solver': ['**liblinear**', 'saga'], 'max_iter': [**100**, 200, 300]
MultinomialNB	'alpha': [0.1, **1**, 10], 'fit_prior': [**True**, False]
KNeighbors	'n_neighbors': [3, **5**, 7], 'weights': ['uniform', '**distance**'], 'algorithm': ['**auto**', 'ball_tree', 'kd_tree', 'brute']
Decision Tree	'criterion': ['**gini**', 'entropy'], 'max_depth': [**None**, 5, 10, 15], 'min_samples_split': [**2**, 5, 10], 'min_samples_leaf': [**1**, 2, 4], 'max_features': ['**auto**', 'sqrt', 'log2']
MLP	'hidden_layer_sizes': [**(10,)**, (50,), (100,)], 'activation': ['relu', '**tanh**'], 'solver': ['**adam**', 'sgd'], 'alpha': [**0.0001**, 0.001, 0.01], 'learning_rate': ['constant', '**adaptive**']

optimization via the GridSearchCV[1] classes.

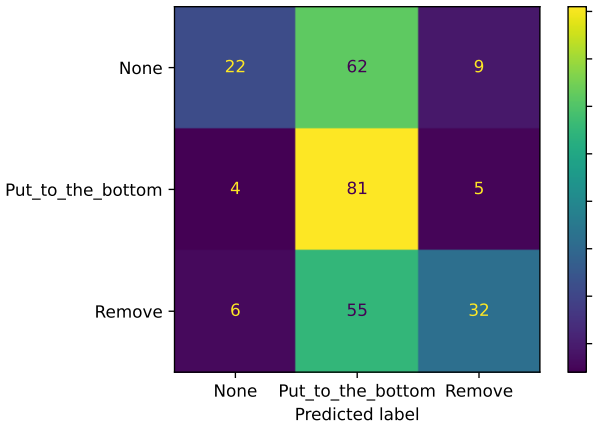

Figure 1: Confusion matrix for the Random Forest model.

4 Results and Discussion

For the ML methods, We used 70% training data and 30% test data by random sampling. We also observed that the dataset was imbalanced (cf. Table 1). To overcome the imbalance issue, we utilized the oversampling method (Chawla et al., 2002). The overall accuracy was used to evaluate the ML methods, but we also computed the per-class accuracy. Table 2 shows the Accuracy, Macro average, and Weighted average of the different ML methods for testing data. From Table 2, the random forest method outperforms with 0.49 accuracy, like the decision tree and multi-layer perceptron method. The linear support vector method, logistic regression, and multinomial naive bayes methods also achieved similar performance with 0.48 accuracy. The confusion matrix (in Fig. 1) also assesses the performance of the random forest method for this experiment. It concisely represents the model's predictions, enabling a detailed analysis of each class's classification accuracy and error rates.

We also processed all the KGs by ABSTAT; since we worked with the public web application, which has a maximum KG upload limit of 10 GB, this reduced the number of KGs. More precisely, the KGs used to analyze the post-processing effect comes from different domains (such as linguistics, COVID-19, etc.) are listed in Table 3. We observe that KGs are very heterogeneous; for instance, there are KGs that barely or do not at all provide types for entities.

Once profiles are computed, ABSTAT returns a set of patterns. Then we applied customizable heuristic post-processing relying on the best ML method (from Table 2). For each ML method, the

[1] scikit-learn-GridSearchCV

Table 4: Patterns before and after post-processing with ML vs. manual patterns (linguistic and COVID-19 KGs).

KG name	Before	Post-processing with ML	Post-processing with manual patterns
basque-eurowordnet-lemon-lexicon-3.0	74	32	47
catalan-eurowordnet-lemon-lexicon-3.0	78	32	47
dbpedia-spotlight-nif-ner-corpu	52	5	37
apertium-rdf-ca-it	15	2	2
wordnet	39	35	36
wn-wiki-instances	4	0	0
asit-data	67	27	52
Reuters-128	21	1	15
lemonwiktionary	19	0	0
apertium-rdf-fr-ca	15	2	0
SimpleEntries	4752	2533	4445
news-100-nif-ner-corpus	21	1	15
drugbank	1408	13	13
pro-sars2	12	0	0
COKG-19-Schema	7	0	0
cord19-akg	108	55	55

post-processing tool provides the options "None", "Put to the bottom" and "Remove", and applies them to the results. For example, Table 4 presents the pattern frequency difference for the KGs upon application of the "Remove" option with the random forest ML method and manual post-processing. The difference is tiny for some KGs, such as WordNet, Drugbank, etc. In contrast, it is quite significant for most others, outliers being SimpleEntries or Asit-data, with much larger reduction obtained using ML than using the manual method. We primarily aimed to reduce the number of patterns in this study; the option "Put to the bottom" is also offered by the ML-based post-processing tool since even patterns containing generic concepts and datatypes can be interesting, particularly for the subsequent detailed scrutiny of a chosen dataset. After familiarizing with the essential nature of a KG, the user may wish to study even such 'de-prioritized' patterns at the bottom of the list.

From Table 4, we can say that, for most of the KGs, with the ML and manual methods, ABSTAT pattern post-processing has a huge impact. Also, ML and manual methods of post-processing have significant differences. The top patterns before and after post-processing are available from an auxiliary page [2].

[2] ABSTAT-patterns-post-processing-with-ML

The post-processing is even more significant for the ML-based approaches than the manual approach, although the number of KGs is too small to make ultimate conclusions. Also, we observed that the dataset that we utilized for the ML methods has a higher effect on (1) KGs with a very low percentage of typing assertions as ABSTAT by default assigns `owl:Thing` as the type for un-typed entities and (2) KGs with a majority of data type relational assertions as many of the elements in the dataset.

5 Conclusions and future work

The experiment suggests that simple heuristics leading to the suppression of patterns containing generic concepts or datatypes might improve the output of state-of-art profiling tools with different ML methods in the context of rapid skimming of multiple KGs.

The present method of training dataset construction primarily relied on manual labeling of the *individual entities* (complemented by whole namespaces, whose pruning is primarily relevant for meta-level vocabularies such as RDF, OWL, or SKOS). However, we are aware that the interestingness of a pattern may be estimated more precisely based on whole pattern triples. We also plan to apply manual labeling at the pattern level. However, the much

larger combinatorial space to be covered will require a significantly increased labor force, possibly recruited via a crowd-sourcing platform.

While the experiment was carried out via a separate ML-based post-processing tool, we will explore how a similar functionality could be achieved within ABSTAT without compromising its current user experience or risking inadequate information loss. Additionally, the dataset utilized by the different ML methods was small; we could also consider enriching the dataset in the future. Also, the generic concepts that occur in many KGs could be eliminated by applying a threshold value on the *inverse KG frequency* (analogous to the common IDF metric).

Acknowledgments

The research was supported by CHIST-ERA within the CIMPLE project (CHIST-ERA-19-XAI-003), and by Nexus Linguarum (COST Action CA18209).

References

Leo Breiman. 2001. Random forests. *Machine learning*, 45:5–32.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Michael P LaValley. 2008. Logistic regression. *Circulation*, 117(18):2395–2399.

Nandana Mihindukulasooriya, María Poveda-Villalón, Raúl García-Castro, and Asunción Gómez-Pérez. 2015. Loupe-an online tool for inspecting datasets in the linked data cloud. *ISWC (Posters & Demos)*, 1:1.

Leif E Peterson. 2009. K-nearest neighbor. *Scholarpedia*, 4(2):1883.

Gollam Rabby, Farhana Keya, Vojtech Svatek, and Renzo Arturo Alva Principe. 2022. Effect of heuristic post-processing on knowledge graph profile patterns: cross-domain study. In *ProLingKnower 2022*. Published on Zenodo.

Hassan Ramchoun, Youssef Ghanou, Mohamed Ettaouil, and Mohammed Amine Janati Idrissi. 2016. Multilayer perceptron: Architecture optimization and training.

S Rasoul Safavian and David Landgrebe. 1991. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674.

Blerina Spahiu, Riccardo Porrini, Matteo Palmonari, Anisa Rula, and Andrea Maurino. 2016. Abstat: ontology-driven linked data summaries with pattern minimalization. In *The Semantic Web: ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29–June 2, 2016, Revised Selected Papers 13*, pages 381–395. Springer.

Shan Suthaharan and Shan Suthaharan. 2016. Support vector machine. *Machine learning models and algorithms for big data classification: thinking with examples for effective learning*, pages 207–235.

Shuo Xu, Yan Li, and Zheng Wang. 2017. Bayesian multinomial naïve bayes classifier to text classification. In *Advanced Multimedia and Ubiquitous Engineering: MUE/FutureTech 2017 11*, pages 347–352. Springer.