Workshop Notes

11th International Workshop

"What can FCA do for Artificial Intelligence?"

FCA4AI 2023

32nd International Joint Conference on Artificial Intelligence

IJCAI 2023

August 20 2023

Macao, S.A.R. China

Editors

Sergei O. Kuznetsov (HSE University Moscow)

Amedeo Napoli (LORIA Nancy)

Sebastian Rudolph (TU Dresden)

http://fca4ai.hse.ru/2023/

IJCAI/2023 MACAO

Preface

The ten preceding editions of the FCA4AI Workshop (see `http://www.fca4ai.hse.ru/`) showed that many researchers working in Artificial Intelligence are deeply interested by a well-founded method for classification and data mining such as Formal Concept Analysis (see `https://upriss.github.io/fca/fca.html`).

The FCA4AI Workshop Series started with ECAI 2012 (Montpellier) and the last edition was co-located with IJCAI-ECAI 2022 (Vienna, Austria). The FCA4AI workshop has now a long history and all proceedings are available as CEUR proceedings (see `http://ceur-ws.org/`, volumes 939, 1058, 1257, 1430, 1703, 2149, 2529, 2729, 2972, and 3233). This year, the workshop has again attracted researchers from different countries working on actual and important topics related to FCA, showing the diversity and the richness of the relations between FCA and AI.

Formal Concept Analysis (FCA) is a mathematically well-founded theory aimed at data analysis and classification. FCA allows one to build a concept lattice and a system of dependencies, i.e., implications and association rules, which can be used for many AI needs, e.g. knowledge discovery, machine learning, knowledge representation and reasoning, natural language and text processing. Recent years have been witnessing increased scientific activity around FCA. In particular an important line of work is aimed at extending the possibilities of FCA w.r.t. data and knowledge processing, and dealing with complex data. These extensions open new directions for AI practitioners. Accordingly, the workshop will investigate the following issues:

- How can FCA support AI activities such as knowledge discovery, knowledge representation and reasoning, machine learning, natural language processing, information retrieval...
- How can FCA be extended for helping AI researchers to solve new and complex problems, in particular how to combine FCA and neural classifiers for allowing interpretability and producing valuable explanations...

First of all we would like to thank all the authors for their contributions and all the PC members for their reviews and their precious collaboration. The papers submitted to the workshop were carefully peer-reviewed by three members of the program committee. The order of the papers in the proceedings (see table of contents in page 5) follows the program of the workshop (see `http://fca4ai.hse.ru/2023/`).

The Workshop Chairs

Sergei O. Kuznetsov
National Research University Higher School of Economics, Moscow, Russia

Amedeo Napoli
Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France

Sebastian Rudolph
Technische Universität Dresden, Germany

Program Committee

Contents

Knowledge Base pattern structures-based Classification of underground forums: A case study

Abdulrahim Ghazal[1]

[1]National Research University Higher School of Economics, Pokrovsky boulevard, 11, 109028, Moscow, Russian Federation

Abstract

Underground forums messages are online platforms where hackers share information and tools for cyber-attacks. This paper discusses using a knowledge base in the context of lazy classification of underground forums messages using pattern structures to assess the risk of these messages. Comparing the performance of pattern structures and the knowledge base approach shows a significant improvement in time needed for classification without loss in accuracy.

Keywords

Formal concept analysis (FCA), Threat intelligence, Underground forums, Pattern structures, Knowledge Bases

1. Introduction

Corporations and organizations have been under increasing level of cyber attacks happening in variety of intensity, frequency and impact [1]. This increase has made collecting information and analyzing findings about these attacks more vital. Threat Intelligence is the practice that focuses on that, and provides insights to victims on the history, current state of the attacks and what to do to mitigate them [2]. The field of Threat Intelligence has been growing a lot in the past years due to the business and regulations needs for more aware cybersecurity practices.

The adequate implementation of such service includes monitoring, detecting, analyzing and reporting cyber threats in a timely manner. Sources of information include public and private underground communities (forums) where the attackers share information about their tools, findings and results.

Underground hackers forums are social platforms that host a group of topics with comments from members of the forum. They consist of sub-forums each focusing on a specific sub-field related to cyber crime. While some of these forums include sections for sales of illegal physical goods, threat intelligence focuses on cyber threats only.

Some underground forums are public (like the forum in Figure 1), but many require registration and in some cases references or payment that ranges from 50-1000. These payments are used sometimes to upgrade a user's status in the forum (VIP, Golden, etc.), and the user would be able to access all sections of the forum. These payments are usually done via cryptocurrency. These forums support "escrow" services with guarantees for deals that are done on the forum.

agazal@hse.ru (A. Ghazal)

Collecting the relevant information is the first step to perform impactful threat intelligence, which is followed by detecting threats inside the collected information, while the threat is fresh, or even better, before it occurs. Such process is usually preformed by human analysts who have to go through a large amount of posted messages daily. This effort grows every day with more forums and more messages.

This work aims to assist the analysts in their task and automate the process of detecting information about threats, giving them more time to move on to the next phases of analysis and reporting threats to the relevant authorities or victims.

This will be done by using natural language analysis of the messages with formal concept analysis and its extension pattern structures to classify messages into risky or none risky. The method used should be fast enough to catch up with the incoming stream of messages, and carry some ability to provide a simple explanation of the result of classification.

In this paper, we focus on using knowledge bases built from previous training iterations of lazy classification using pattern structures, and comparing the knowledge base with the raw application of pattern structures lazy classification.

The rest of the paper is organized as follows: In Section 2 we recall basic definitions in formal concept analysis, pattern structures and lazy classification method using pattern structures. In Section 3 we describe the experimental setting and knowledge base building. In Section 4, we discuss the preliminary results of applying the knowledge base approach in the context of lazy classification to underground forum messages. We conclude the work in section 5.

2. Formal Concept Analysis

2.1. Main Definitions

Formal Concept Analysis (FCA) is a mathematical theory that is based on concepts and conceptual hierarchy [3, 4]. Its structuring of knowledge representation was used for knowledge discovery, data analysis [5, 6], mining association rules [5, 7] ontology design [8, 9], and recommendation systems [10].

2.2. Pattern Structures

This extension of Formal Concept Analysis was developed as an effort to enable applying the mathematical tools offered by Formal Concept Analysis with a more complex data structures like graphs, non-binary or vector data [11].

Let G be a set of objects and (D, $\sqcap$) be a meet-semi-lattice of possible object descriptions or patterns (for standard FCA, it would be the powerset of attribute set) with the similarity operator $\sqcap$. Elements of D are ordered by a subsumption relation $\sqsubseteq$ such that a,$b \in D$, then one has $a \sqsubseteq b \Leftrightarrow a \sqcap b = a$. We also define $\delta : G \rightarrow D$ as a mapping between objects and their attributes. We call (G, $\underline{D}$, δ) where $\underline{D}$ = (D, $\sqcap$) a pattern structure. We can define the operators $(\cdot)^{\diamond}$ on $A \subseteq G$ and $d \in (D, \sqcap)$ making Galois connection between the powerset of objects and ordered set of descriptions:

$$A^{\diamond} = \sqcap_{g \in A} \delta(g) \quad (1)$$

```
Algorithm 1: Lazy Classification with Pattern Structures
Requires: pattern structure $(G, \underline{D}, \delta)$, test example $g_t \in G_\tau$ with description $\delta(g_t)$,
parameter $0 \leq \alpha \leq 1$.
1: for $g \in G_+ \cup G_-$ :
2: compute sim = $\delta(g) \sqcap \delta(g_t)$
3: extsim = $(\text{sim})^\diamond$
4:        if $\alpha\%$ of objects in extsim have target attribute, classify $g$ positive
5:        if $\alpha\%$ of objects in extsim do not have target attribute, classify $g$ negative
6: classify undetermined (the algorithm terminates without classification).
```

$$d^\diamond = \{g \in G \mid d \sqsubseteq \delta(g)\} \tag{2}$$

These operators will give us back the maximal set of patterns shared by the objects in A and the maximal set of objects that share the description d, respectively.

A pair (A, d), $A \in G$ and $d \in (D, \sqcap)$ that satisfies $A^\diamond = d$ and $d^\diamond = A$ is called a *pattern concept*, where A is called the *extent* and d is called the *pattern intent* of (A, d).

A partial order $\leq$ is defined on the set of concepts: $(A, d_1) \leq (B, d_2)$ iff $A \subseteq B$ (or, equivalently, $d_2 \sqsubseteq d_1$). This partial order forms a complete lattice on the set of all pattern concepts. We call this the pattern concept lattice of the pattern structure $(G, \underline{D}, \delta)$.

For classification tasks we do not need to extract the full hidden knowledge from a dataset in terms of implications, hypotheses or association rules, but a so-called lazy classification can be applied [12, 13].

2.3. Lazy Classification with Pattern Structures

In classification problems we have a target attribute, which, in the simplest case of two classes, has two values, denoted by $+$ and $-$. By G_+ we denote the set of objects that have the target attribute (positive examples) and by G_- we denote the set of objects that do not have the target attribute (negative examples), so that $G_+ \cap G_- = \emptyset$. Elements of G that do not belong to any of these subsets are called unclassified examples G_τ.

A version of the lazy classification method [12, 13] is described in Algorithm 1.

This algorithm takes $O(|\mathrm{G}|\ (p(\sqcap) + |\mathrm{G}|\ p(\sqsubseteq)))$ time, where $p(\sqcap)$, $p(\sqsubseteq)$ are times for computing $\sqcap$, $\sqsubseteq$, respectively.

2.4. Knowledge Bases

With the increase of data sizes used to perform some information retrieval or computation on the stored data, it becomes challenging to generate results in a timely manner, which led to the creation of knowledge bases that store some previously proven information that can help in these tasks for fast access [14].

In the context of FCA and pattern structures, concept lattices can offer a new method of knowledge representation [15] and in this work, it will be applied to save time by testing new objects versus well-performing classifiers that are saved from past testing iterations first.

3. Experiments

The goal of experiments is to test how using a knowledge base will change the results of the raw application of the pattern structures lazy classification scheme (both in time and performance). To do that, we will need to test several settings of the pattern structures lazy classification scheme with different parameters, then repeat the best performing experiments, but with using the knowledge base instead.

We will perform several experiments starting with lazy classification using the traditional FCA approach, then use the interval, min and max pattern structures. We will repeat the same experiments, but with probabilistic relaxation. In the end, we will repeat the experiments with best performing parameters, but with the use of the knowledge base, to measure the improvement in time needed for classification.

3.1. Dataset

The used dataset is composed of text messages posted by hackers on several underground forums starting from 2021 Provided by the cybersecurity firm F.A.A.C.T. The positive examples of the dataset are real threats detected by human analysts and reported upon. The dataset is balanced in terms of classes and has 4945 messages. These messages come from a core of real threat messages that were selected by the Threat Intelligence analysts team in the aforementioned company. The negative sample was obtained from the same set of underground forums which contained the positive samples, and posted in the same time frame.

The dataset is then processed into a numerical dataset for the later stages, with the use of tf-idf. We should note that the number of keywords that will be included in the results of tf-idf is a parameter that will be controlled during the experiments and is called “min_df”. It represents the threshold of percentage of documents at which a keyword is included in the vocabulary. We will also control the tolerance factor α which represents the probabilistic relaxation allowed for counter examples (See Algorithm 1).

3.2. Assessment

we should note that the used version of the lazy classification algorithm allows for unclassified cases, and in this spirit, we define “**Saved Effort**” measure, which is computed as $1 - \frac{|G_{uncl}|}{|G_\tau|}$, where G_{uncl} is the set of unclassified examples $G_{uncl} \subseteq G_\tau$.

3.3. Experiments

In all the following experiments, 5-cross validation was used. The code used to perform these experiments is written in python 3.6 and no Off-The-Shelf tools were used to write the FCA or Pattern Structures code, but sk-learn package was used to build the Machine Learning models in the later sections. Due to space limitations, we present the numerical results (publicly accessible) at: https://github.com/abdulrahimGhazal/FCA-LC-KB.

3.3.1. Binary Attributes

The attribute values here are the tf-idf values for the keywords contained in the text that were included in the vectorizer's vocabulary resulting from tf-idf. It would be represented as:

$$att\_value(keyword) = \begin{cases} 1 & keyword \in vectorizer\ vocab \quad (3) \\ 0 & otherwise \quad (4) \end{cases}$$

We tested 5 values of min_df and the highest F1 value was 0.98 and saved effort at 0.88 with min_df at 0.01. We repeat the same experiment, but with introducing the tolerance factor, allowing for a small amount of counter examples. We get the highest F1 value 0.98 and saved effort 0.92 with min_df at 0.01, and α at 75%.

3.3.2. Interval Pattern Structure

We represent the values of tf-idf as intervals of the floating point value, such that if the tf-idf value for a keyword is x, then the attribute value would be an interval $[x,x]$. the intersection operator for this pattern structure is defined as:

$$[a_1, b_1] \sqcap [a_2, b_2] = [min(a_1, a_2), max[b_1, b_2)] \quad (5)$$

We tested 5 values of min_df and the highest F1 value was 0.88 and saved effort at 0.88 with min_df at 0.01. We repeat the same experiment, but with introducing the tolerance factor, allowing for a small amount of counter examples. We get the highest F1 value 0.94 and saved effort 0.94 with min_df at 0.01, and α at 75%.

3.3.3. Min Pattern Structure

We represent the values of tf-idf as intervals of the floating point value, such that if the tf-idf value for a keyword is x, then the attribute value would be an interval $[x, \infty[$. the intersection operator for this pattern structure is defined as:

$$[a_1, \infty] \sqcap [a_2, \infty] = [min(a_1, a_2), \infty] \quad (6)$$

We tested 5 values of min_df and the highest F1 value was 0.90 and saved effort at 0.87 with min_df at 0.01. We repeat the same experiment, but with introducing the tolerance factor, allowing for a small amount of counter examples. We get the highest F1 value 0.94 and saved effort 0.94 with min_df at 0.01, and α at 75%.

3.3.4. Max Pattern Structure

We represent the values of tf-idf as intervals of the floating point value, such that if the tf-idf value for a keyword is x, then the attribute value would be an interval $]-\infty, x[$. the intersection operator for this pattern structure is defined as:

$$]-\infty, a_1] \sqcap]-\infty, a_2] =]-\infty, max(a_1, a_2)] \quad (7)$$

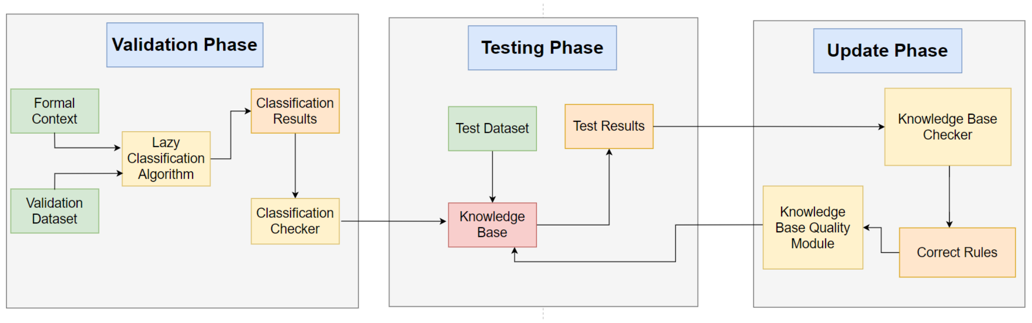


Figure 1: Knowledge Base building and updating.

We tested 5 values of min_df and the highest F1 value was 0.88 and saved effort at 0.88 with min_df at 0.01. We repeat the same experiment, but with introducing the tolerance factor, allowing for a small amount of counter examples. We get the highest F1 value 0.94 and saved effort 0.94 with min_df at 0.01, and α at 75% and 80%.

3.3.5. knowledge base experiments

After noticing that a lot of examples are classified using a limited set of attribute sets (from the intersection of new example and randomly selected examples from the objects set), so a knowledge base from all "classifiers" which are basically sets of attributes, that performed well was created. A diagram of the process of building and maintaining the knowledge base can be seen in Figure 1.

This knowledge base was then checked manually by human analysts (experts) to review whether the attributes really carried some truth in the classification. We set the threshold for adding the classifier to the knowledge base as that it must classify correctly 90% of the test examples assigned to it, and it must classify at least 2% of the test examples.

The building of the knowledge base is done via using a validation dataset that is part of the data, with a 5 cross validation process. the classification results then are given to the "classification checker" module, which checks if a classifier was able to classify correctly. If that was the case, the classifier would be added to the interim knowledge base which would be checked in the end of the validation process to trim the classifiers that do not match our predefined conditions.

In case we do not want to update the knowledge base, The resulting classifiers (ready knowledge base) will be passed on to the test dataset, which the classifiers never saw before to avoid over-fitting. If the results are satisfactory, we stop the process and save the knowledge base to use in real-world cases.

If an update is needed, we also created a monitoring module to update the knowledge base, after each iteration of testing, so that these conditions are not broken by old classifiers, and in the same time, we should maintain the time advantage that the knowledge base with a specific set of classifiers have. The process of updates pass the ready knowledge base to the "knowledge base checker" module which checks if there are any changes in the rules from the old version of the knowledge base, and keeps lists of the removed rules and the added rules.

These rules are then passed to the "knowledge base quality module" which works by making several intermediate steps towards minimizing the changes while keeping the thresholds of quality. These intermediate steps would mean breaking some of the quality thresholds, but only for the removed rules, and only in case the threshold of the number of classified examples (2%) is not achieved. This is due to the new test dataset. The resulting rules then make the final updated knowledge base.

Binary Rules In the following experiments, we only depend on the existence of the set of attributes as a classifier, so when using the knowledge base, we only check if the message contains the set of attributes in its text.

Looking at the results, while the best time was observed in traditional FCA before using the knowledge base and after using it at 0.03 milliseconds on average, we notice that the time needed for the rest of tested pattern structures has decreased significantly from more than one second to 0.04 milliseconds, with a slight loss in performance.

Conditioned Rules Unlike the previous experiment, we save the values of the attributes in the set of the attributes of the to-be-added classifiers, then we apply the intersection of the corresponding pattern structure on the whole test cases.

Since the binary attributes already mean that the keyword exists or not, the results will not change for the binary attributes experiments. For the pattern structure experiments, we notice that the loss of accuracy has decreased, but the time needed has increased slightly, since we have to check more rules before reaching a classification, but still much better than going through the usual lazy classification scheme.

While there is not a large difference between the knowledge base approaches, there is a large time improvement in comparison to the traditional pattern structures or binary FCA approaches.

3.3.6. Other ML models experiments

We trained several machine learning models with the dataset we have to observe how our work measures to common classification models. The experiments were run two times, one with binarized attributes and another with floating-point values of the tf-idf model. The models used include:

- Decision Trees
- Gaussian Naive Bayes
- Support Vector Machines
- Logistic Regression
- Random Forests

The results show a close performance in relation to the pattern structures, which was the SVM model in the case of binary attributes with an F1 value of 0.96 but less time needed to reach a classification, with the best average time in the case of Decision Trees model with floating-point values with 0.008 seconds. The times needed in most of the models used were better than the experiments with the knowledge base.

Some of these models are rules-based, and others are model-based, but in all of these models, there is not a possibility for explaining the results simply. This can be simply done by returning the attribute intersection that led to the classification if the lazy classification algorithm was run, or the classifier in case of the knowledge base experiments.

4. Discussion

The results of the experiments show that the best performance in terms of F1 score was given by the binarized attributes. The problem with such methods is their flexibility to perform well when new data is presented.

When pattern structures are used, the less restrictive the values representation and intersection operator, the better it performs. Thus, the best pattern structure was the Min pattern structure, followed by the Max and finally the Interval pattern structure.

The introduction of knowledge bases will save a lot of time searching for the best fit of intersected attributes that could produce a classification, but other questions need to be addressed, like how often the knowledge base must be updated, and how to ensure that there is no bias due to the old data. It is worth noting that building the knowledge base while keeping the floating-point values of the attributes and applying the intersection operator to include (or exclude) ranges of values that might give incorrect results in general, but be locally successful gives better accuracy but needs more time, so a trade-off is established.

While the time needed for common machine learning models is less in most cases, this presents an opportunity for improving the implementation of the knowledge base classification, since the implementation of the used machine learning models (sk-learn) applies multi-threading when possible, which speeds up the classification.

The metrics used to assess such kind of experiments also come to discussion, as the nature of the data and the methods give rise to issues when using F1 for example instead of recall, as in such cases, where the cost of getting Type I errors is high.

5. Conclusion

We presented a knowledge base approach for lazy classification using pattern structures of underground forums messages, and the results of the use of the knowledge base are promising in terms of saved time needed to reach a classification.

6. Acknowledgements

The article was prepared within the framework of the Basic Research Program at HSE University, RF

References

[1] C. P. R. Team, Check point research reports a 38% increase in 2022 global cyberattacks, 2023. URL: https://blog.checkpoint.com/2023/01/05/38-increase-in-2022-global-cyberattacks/.

[2] R. Future, The Threat Intelligence Handbook: A Practical Guide for Security Teams to Unlocking the Power of Intelligence, 1st. ed., CyberEdge Group, 2018.

[3] B. Ganter, R. Wille, Formal concept analysis: mathematical foundations, Springer Science & Business Media, 2012.

[4] S. Ferré, M. Huchard, M. Kaytoue, S. O. Kuznetsov, A. Napoli, Formal Concept Analysis: From Knowledge Discovery to Knowledge Processing, Springer International Publishing, Cham, 2020, pp. 411–445. URL: https://doi.org/10.1007/978-3-030-06167-8_13. doi:10.1007/978-3-030-06167-8_13.

[5] M. Kaytoue, S. O. Kuznetsov, A. Napoli, S. Duplessis, Mining gene expression data with pattern structures in formal concept analysis, Information Sciences 181 (2011) 1989–2001.

[6] A. Masyutin, Y. Kashnitsky, S. O. Kuznetsov, Lazy classication with interval pattern structures: Application to credit scoring, in: FCA4AI@ IJCAI, 2015.

[7] W. Saidi, Formal concept analysis based association rules extraction (2012) 490–497.

[8] M. Obitko, V. Snasel, J. Smid, V. Snasel, Ontology design with formal concept analysis., in: CLA, volume 128, 2004, pp. 1377–1390.

[9] G. Jiang, K. Ogasawara, A. Endoh, T. Sakurai, Context-based ontology building support in clinical domains using formal concept analysis, International journal of medical informatics 71 (2003) 71–81.

[10] P. Vilakone, K. Xinchang, D.-S. Park, Movie recommendation system based on users' personal information and movies rated using the method of k-clique and normalized discounted cumulative gain, Journal of Information Processing Systems 16 (2020) 494–507.

[11] B. Ganter, S. O. Kuznetsov, Pattern structures and their projections, in: International conference on conceptual structures, Springer, 2001, pp. 129–142.

[12] S. O. Kuznetsov, Scalable knowledge discovery in complex data with pattern structures, in: International Conference on Pattern Recognition and Machine Intelligence, Springer, 2013, pp. 30–39.

[13] S. O. Kuznetsov, Fitting pattern structures to knowledge discovery in big data, in: International conference on formal concept analysis, Springer, 2013, pp. 254–266.

[14] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Pearson, 2021.

[15] K. E. Wolff, A conceptual view of knowledge bases in rough set theory, in: International Conference on Rough Sets and Current Trends in Computing, 2001, p. 220–228.

Interval pattern structures for interpreting K-nearest neighbor approach in lazy classification

Position Paper

Alan Tomat[1]

[1]Moscow Institute of Physics and Technology, Institutskiy per. 9, Dolgoprudny, Moscow Region 141700, Russia

Abstract

This paper proposes IPS-KNN, an interpretable variant of the distance-weighted K-nearest neighbors (KNN) algorithm based on interval pattern structures, to address the limitation of KNN's lack of interpretability. The proposed algorithm provides a reason for the classification in the form of a pattern of the interval pattern structure describing the dataset. The intervals in the reason for classification provide insights into which features are important for the classification task and what values these features should have to produce a specific classification result. The IPS-KNN algorithm was evaluated on the red wine quality dataset, where it performed similarly to the distance-weighted KNN algorithm in terms of classification performance. The proposed algorithm can be used in applications where interpretability is important, such as in medical diagnosis or credit risk assessment.

Keywords

lazy classification, pattern structures, interval pattern structures, interpretable K-nearest neighbors

1. Introduction

Lazy learning, also known as instance-based learning, is a type of machine learning algorithm that does not explicitly train the model. Instead, it saves all the training data and uses a similarity measure between the training data and the new data to make predictions [1]. The term "Lazy learning" was first introduced in 1991 in [2]. It was defined as a type of machine learning algorithms that postpones the training process to the testing phase. That is until a new instance is provided for classification. The majority of lazy classification algorithms are K-Nearest Neighbors (KNN) algorithms [3]. The results of KNN algorithms are not inherently interpretable. When both K and the number of features in the dataset are small, visualizing the K-nearest neighbors can provide some sense of interpretability, but not a formal one.

The interpretability of machine learning models lacks a precise mathematical definition. However, there are several non-mathematical definitions that have been proposed. One such definition, stated by Miller in [4], is that interpretability refers to "the degree to which a human can understand the cause of a decision". The researchers in [5] defined interpretable models as those for which humans can understand the causes of the model's predictions.

FCA4AI eleventh edition: What can FCA do for Artificial Intelligence?, August 20, 2023, Moscow, Russia

alantomat1998@gmail.com (A. Tomat)
0009-0005-6641-2870 (A. Tomat)

While interpretability may not be crucial in low-risk applications such as recommendation systems, it is of utmost importance in high-risk applications such as credit risk assessment and medical diagnosis. In such cases, it is essential to pair each prediction with the reason that led the model to make that prediction in order to avoid any possible errors. Despite the absence of a standardized metric for evaluating interpretability, models can be compared based on how easily humans can comprehend the reasons behind their decisions [6].

Pattern structures, an extension of Formal Concept Analysis (FCA) introduced by Kuznetsov and Ganter [7], expands on FCA by enabling the analysis of complex descriptors. Pattern structures, among other generalizations of FCA, have been widely adopted in diverse applications such as information retrieval, web and ontology engineering, biclustering and recommendation, databases and functional dependencies, and software engineering [8].

Interval pattern structures, proposed in [9], are a special case of pattern structures in which the descriptors are numeric intervals. They enable the use of FCA-based knowledge discovery tools on numeric datasets [10]. To classify new instances, pattern structures can be used to extract hypotheses from a set of training instances [11]. The original hypotheses-based lazy classification algorithm on pattern structures was proposed in [12]. This algorithm finds the similarity between a query instance and each instance in the training set, and considers a similarity to be a hypothesis only if it describes instances from a single class. The extracted hypotheses describe the characteristics that the new instance shares with a subset of instances from the training set, providing interpretability into the model's predictions.

Although the original algorithm was proven effective on pattern structures with graph descriptors [11], it suffered from too specific hypotheses in the case of numeric features, especially when the number of features increased [13]. Too specific hypotheses describe only few instances and do not convey much useful information for the classification process. To address this issue, [13] proposed randomly sampling batches of instances from each class in the training set and finding the similarity between the query instance and the entire batch.

Instead of attempting to solve the issues with the original hypotheses-based algorithm, this paper argues that augmenting the distance-weighted KNN with interval pattern structures can provide interpretability. As the nearest instances to the query instance carry the most useful information for classification, this approach focuses on leveraging this information, rather than relying on random batches from the train set.

2. Formal Definitions of Pattern Structures and Interval Pattern Structures

2.1. Pattern Structures

Pattern structures are a generalization of Formal Concept Analysis (FCA). In FCA, a context is defined as a set of instances with binary features, where a binary relation specifies which features are possessed by each instance [14]. Pattern structures remove the binary features condition and allow the use of any type of features as long as they form a lower semi-lattice. We follow [7] to provide a formal definition: Let G be a set of instances, $(D, \sqcap)$ a lower semi-lattice of all possible instance descriptors, and $\delta : G \rightarrow D$ a mapping that corresponds each instance

$g \in G$ with its descriptor $d \in D$.

Then $(G, \underline{D}, \delta)$ is called a *pattern structure*, where $\underline{D} = (D, \sqcap)$, with the condition that the set $\delta(G) := \{\delta(g) | g \in G\}$ generates a complete sub-semi-lattice $(D_\delta, \sqcap)$ from $(D, \sqcap)$, i.e. each subset E of $\delta(G)$ has a meet $\sqcap E$ in $(D, \sqcap)$.

The elements of D are called *patterns* and are naturally ordered by the *absorption* relation $\sqsubseteq$: for $c, d \in D$, $c \sqsubseteq d \iff c \sqcap d = c$. The operation $\sqcap$ is also called the *similarity operation*.

The pattern structure $(G, \underline{D}, \delta)$ follows FCA and defines two *derivative operators* $(.)^\diamond$ given in equations 1 and 2. The first of which is applied to a set of instances and returns the largest common pattern describing these instances, while the second is applied to a pattern and returns a set of instances that possess this pattern.

$$A^\diamond = \sqcap_{g \in A} \delta(g) \quad \text{for} \quad A \subseteq G \tag{1}$$

$$d^\diamond = \{g \in G | d \sqsubseteq \delta(g)\} \quad \text{for} \quad d \in (D, \sqcap) \tag{2}$$

2.2. Interval Pattern Structures

Interval pattern structures [9, 10] are a type of pattern structures in which the descriptors of the set D are represented as p-dimensional vectors of intervals, with each interval corresponding to the value of one feature. Here, p is the number of features being analyzed. Interval pattern structures can be used to represent numeric features, by considering each numeric value x as a zero-length interval $[x, x]$.

Thus, the patterns of the set D are $p-$dimensional vectors of intervals, where p is the number of features. For the patterns $e = \langle [a_i, b_i] \rangle_{i=1,2,..,p}$ and $f = \langle [c_i, d_i] \rangle_{i=1,2,..,p}$ in D, the similarity of e and f is given by:

$$e \sqcap f = \langle [a_i, b_i] \rangle_{i=1,2,..,p} \sqcap \langle [c_i, d_i] \rangle_{i=1,2,..,p} = \langle [min(a_i, c_i), max(b_i, d_i)] \rangle_{i=1,2,..,p} \tag{3}$$

The absorption relation $\sqsubseteq$ is given by:

$$\begin{aligned} e \sqsubseteq f &\iff \langle [a_i, b_i] \rangle_{i=1,2,..,p} \sqsubseteq \langle [c_i, d_i] \rangle_{i=1,2,..,p} \\ &\iff [a_i, b_i] \sqcap [c_i, d_i] = [a_i, b_i] \;\; \forall i \in \{1, 2, .., p\} \end{aligned} \tag{4}$$

Consequently, larger intervals are absorbed by the smaller intervals they contain. At first glance, this may seem counter-intuitive, but by definition, the similarity of a set of instances should be absorbed by the pattern of each instance in this set.

3. KNN-based Interval Pattern Structure Lazy Classifier

The original distance-weighted K-nearest neighbors (KNN) algorithm [1] classifies a query instance by finding its K-nearest neighbors and then computing a score for each class based on the inverse of the distance between the query instance and the instances from the K-nearest neighbors that belong to that class. However, the results of KNN algorithms are not inherently interpretable. To address this limitation, we propose IPS-KNN (Interval Pattern Structure K-Nearest Neighbor): an interpretable variant of the distance-weighted KNN algorithm that is based on interval pattern structures.

3.1. IPS-KNN Lazy Classifier

To classify a new instance g_τ, the set $G_{knn} \subseteq G$ consisting of the k-nearest neighbors of g_τ in the training set G is identified. The similarity of all instances in G_{knn} is then calculated by applying the similarity operator $\sqcap$. All instances in the training set that are described by this similarity are included in a voting process. The class of g_τ is then determined by the function:

$$y(g_\tau) = \mathrm{sgn}(s_+ - s_-) \tag{5}$$

where the positive and negative scores, s_+ and s_- respectively, are calculated as:

$$s_+ = \sum_{g_+ \in G_{knn}^{\diamond\diamond} \cap G_+} \frac{1}{d(g_+, g_\tau)} \tag{6}$$

$$s_- = \sum_{g_- \in G_{knn}^{\diamond\diamond} \cap G_-} \frac{1}{d(g_-, g_\tau)} \tag{7}$$

Here, $d(a, b)$ is the Euclidean distance between instances a and b in the feature space. The set $G_{knn}^{\diamond\diamond}$ is the subset of G that contains all instances described by the similarity $G_{knn}^{\diamond}$.

The proposed IPS-KNN algorithm differs from the original distance-weighted k-Nearest Neighbors in that, in IPS-KNN, the similarity of the k-Nearest Neighbors defines which instances will vote, while in KNN, only the k-nearest neighbors vote. The set of voters in IPS-KNN contains all the k-nearest neighbors because they are all described by their similarity, but it may include additional instances as well.

The difference in the set of voters between IPS-KNN and KNN is due to the geometric shape that encloses the neighbors in the feature space. In KNN, the distance between the query instance and the farthest instance of the k-Nearest Neighbors defines a hyper-sphere in the feature space, and all the instances located on its surface or inside it vote. On the other hand, the similarity of the k-Nearest Neighbors of the query instance defines a hyper-rectangle in the feature space in IPS-KNN. This hyper-rectangle is larger than the hyper-sphere of KNN and contains it entirely within its boundaries. Therefore, the set of voters in IPS-KNN may include more instances than just the k-Nearest Neighbors.

3.2. IPS-KNN Interpretability

The similarity operation $\sqcap$ can reveal why a query instance was classified in a certain class, as it captures the similarity of descriptions. Suppose that a query instance g_τ was classified as positive by IPS-KNN. In this case, $G_{knn}(g_\tau)$ is the set of k-nearest instances to g_τ in the training set, and their common similarity determines the set of voters $V(g_\tau) = G_{knn}^{\diamond\diamond}(g_\tau)$, which can be split into positive voters V_+ and negative voters V_-. The negative score is then given by $s_- = \sum_{g_- \in V_-} \frac{1}{d(g_-, g_\tau)}$. Since g_τ was classified as positive, the positive score s_+ is greater than s_-. It is possible that only a subset of V_+ needs to vote to classify g_τ as positive; any subset of V_+ that produces a positive score greater than the negative score will lead to the same classification result.

Formally, let $F = \{E \mid E \subseteq V_+,\ \sum_{g_+ \in V_+} \frac{1}{d(g_+, g_\tau)} > s_-\}$ be the set of all subsets of $V_+(g_\tau)$ that give a positive score greater than s_-. The subset with the minimum number of instances includes the instances of V_+ closest to the query instance g_τ. However, this subset is not necessarily unique, since multiple instances might have the same distance from the query instance. Let E be the subset that correspond to the hyper-rectangle with minimum volume. The similarity of the elements of E together with the description of the query instance is the reason for classifying g_τ as positive. This is given by $R = E^\diamond \sqcap \delta(g_\tau) \ : \ E \in F \ \text{ and } \ |E| = \min |B| : B \in F \ \text{ and } \ \Pi_{i=1}^{p} |b_i - a_i|_i$ is minimum, where a_i and b_i are the limits of the i_{th} interval of $E^\diamond \sqcap \delta(g_\tau)$ and p is the number of features. The reason for classification R is itself a pattern in the interval pattern structure $(G, \underline{D}, \delta)$, consisting of a vector of intervals with one interval for each feature. The values of these intervals, relative to the original range of values of the features, provide interpretability. R can also be considered a classifier that correctly classifies all instances of E and the new instance g_τ. It should be noted that substituting the subset E with any other element in F would provide another correct interpretable classifier. However, using E corresponds to enclosing the classified instances in the most specific hyper-rectangle. On the other hand, using V_+ instead of E will result in the most general hyper-rectangle.

The interpretability provided by IPS-KNN is somewhat analogous to that provided by decision trees for continuous features. In IPS-KNN, the reason for classification is represented by a hyper-rectangle in the feature space, with one interval for each feature. Similarly, in decision trees, decisions are based on comparisons of feature values using greater than and less than conditions [15], which can also be represented as hyper-rectangles, with two exceptions. Firstly, decision trees may not use all features in the classification process, resulting in intervals of the form $(-\infty, +\infty)$. Secondly, decision trees may bound the values of some features from one side only, leading to intervals of the form $(-\infty, x)$ and $(x, +\infty)$, where x is a real value. Section 4 further explains how these intervals are used for interpretability.

4. Experiments

The proposed algorithm IPS-KNN was evaluated against the baseline classification algorithms on the red wine quality dataset [1], which contains 11 numeric features and its output feature was binarized for this purpose. In its original form, the quality of each wine is assessed by a score between 1 and 10, where a higher score indicates better quality. To binarize the output feature, wines with a quality score in the range $[1, 5]$ were considered 'bad', while those in the range $[6, 10]$ were considered 'good'. To assess the performance of the models, 20% of the instances in the dataset were randomly selected as a holdout test set, and the remaining 80% were used as the training set. In addition, 5-fold stratified cross-validation was used to evaluate the models and to tune their hyperparameters using a grid search approach. F1 score was used as the evaluation metric to take class imbalance into account. The accuracy and F1 scores on the held-out test set for the proposed algorithm, distance-weighted KNN, and other baseline classification algorithms are presented in Table 1. The evaluation results indicate that the IPS-KNN algorithm and the distance-weighted KNN algorithm had similar performance. This suggests that our proposed modification to the KNN algorithm, which allows more instances to vote, did not significantly degrade performance. This is because the K nearest neighbors, which have the largest voting weight, are still included in the set of voters. Table 1 also shows

[1]https://archive.ics.uci.edu/ml/datasets/wine+quality

Table 1
Accuracy and F1 for IPS-KNN and the baseline classification algorithms on the binarized red wine quality dataset.

Classifier	Accuracy	F1
KNN	82.81%	83.97%
IPS-KNN	81.56%	82.8%
Naive Bayes (GaussianNB)	75%	76.05%
Logistic Regression	77.5%	78.44%
SVM	78.75%	80.68%
Decision Tree	77.5%	78.95%
Random Forest	84.06%	84.59%
XGBoost	84.69%	86.04%

that Random Forest and XGBoost outperformed IPS-KNN, with XGBoost achieving the highest F1 score with a 3.26% increase over that of IPS-KNN. Both XGBoost and Random Forest use decision trees, which means that their results can be interpreted using the tests performed at the nodes along the paths through the structure used to classify a query instance. However, after grid search, the number of decision trees used by Random Forest and XGBoost was 100 and 1000, respectively. This large number of decision trees makes direct interpretability a difficult task.

The following is an example of how IPS-KNN classifies a new instance and provides interpretability on the binary red wine quality dataset. Table 2 shows the features of the red wine quality dataset, the values of these features for the query instance g_τ, the ranges of the features, and the intervals of the reason of classification R. We obtained the value of the hyperparameter K of 25 through grid search. The similarity of 25-nearest instances to g_τ chose 44 instances to vote, 38 of which are positive and the remaining 6 are negative. The resulting positive and negative scores were $s_+ = 19.3$ and $s_- = 2.6$; therefore, g_τ was classified as positive.

Out of the 38 positive instances, the nearest 4 were enough to give a positive score larger than the negative one. The similarity of these 4 objects together with g_τ produced the reason of classification denoted by R in Table 2. The intervals of R provide the interpretability of why g_τ was classified as positive and what values should the features of instances similar to g_τ also have in order to be classified as positive.

For example, the interval $[6.3, 6.8]$ of R corresponding to the feature (fixed acidity) shows that this feature's value should be low relative to the original range of values in order to the classification result to be positive. The interval corresponding to the feature (alcohol) shows that the value of this feature should be in in the middle of the range of possible values. The intervals corresponding to the features (fixed acidity, residual sugar, sulphates) are short relative to the ranges of their values, which mean that they are important to the classification process. In contrast, intervals covering a large part of the entire range of values are less useful since they describe almost all objects in the dataset.

Table 2
Ranges of data set feature values (red wine quality) and interval classifier intervals.

Feature	Feature values of g_τ	Range of values	Reason of classification R
Fixed acidity	6.3	$[4.6, 15.9]$	$[6.3, 6.8]$
Volatile acidity	0.55	$[0.1, 1.6]$	$[0.49, 0.67]$
Citric acid	0.15	$[0, 1]$	$[0.02, 0.22]$
Residual sugar	1.8	$[0.9, 15.5]$	$[1.8, 2.3]$
Chlorides	0.077	$[0.01, 0.61]$	$[0.061, 0.077]$
Free sulfur dioxide	26	$[1, 72]$	$[13.0, 37.0]$
Total sulfur dioxide	35	$[6, 289]$	$[24.0, 53.0]$
Density	0.99314	$[0.99, 1.004]$	$[0.99314, 0.99489]$
pH	3.32	$[2.7, 4]$	$[3.32, 3.41]$
Sulfates	0.82	$[0.3, 2]$	$[0.76, 0.83]$
Alcohol	11.63	$[8.4, 14.9]$	$[10.3, 11.6]$

5. Conclusion

Based on the results of the experiments, we can conclude that the IPS-KNN algorithm performs similarly to the distance-weighted KNN algorithm in terms of classification performance. However, the IPS-KNN algorithm provides additional interpretability through the reason for the classification in the form of a pattern of an interval pattern structure. The intervals in the reason for classification provide insights into which features are important for the classification task and what values these features should have in order to produce a specific classification result.

The proposed algorithm can be used in applications where interpretability is important, such as in medical diagnosis or credit risk assessment. The interpretability provided by IPS-KNN can help experts understand why a certain decision was made, which can lead to better decision making and increased trust in the system.

6. Future Work

One limitation of the IPS-KNN algorithm is that the hyper-rectangle surrounding the query instance and its neighbors is currently aligned with the coordinate axes in the feature space. To address this limitation, we plan to investigate the introduction of a tilted hyper-rectangle in future work. By tilting the hyper-rectangle at an angle, we aim to provide better separation of the voters and potentially improve classification accuracy.

Introducing a tilted hyper-rectangle is a promising avenue for future research that could enhance the performance of the IPS-KNN algorithm. However, it is important to conduct more detailed investigation to explore the feasibility and potential benefits of this approach. Specifically, we need to determine the optimal angle of tilt and evaluate the impact of this approach on classification accuracy compared to the current IPS-KNN algorithm.

References

[1] D. W. Aha, D. Kibler, M. K. Albert, Instance-based learning algorithms, Machine learning 6 (1991) 37–66.

[2] D. G. Lowe, Learning in Autonomous Agents: Exploration, Curiosity, and Learning in the Absence of External Rewards, Ph.D. thesis, Department of Computer Science, University of British Columbia, 1991.

[3] D. Wettschereck, D. W. Aha, T. Mohri, A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms, Artificial Intelligence Review 11 (1997) 273–314.

[4] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial intelligence 267 (2019) 1–38.

[5] E. Dudyrev, I. Semenkov, S. O. Kuznetsov, G. Gusev, A. Sharp, O. S. Pianykh, Human knowledge models: Learning applied knowledge from the data, Plos one 17 (2022) e0275814.

[6] C. Molnar, Interpretable machine learning, Lulu. com, 2020.

[7] B. Ganter, S. O. Kuznetsov, Pattern structures and their projections, in: Conceptual Structures: Broadening the Base: 9th International Conference on Conceptual Structures, ICCS 2001 Stanford, CA, USA, July 30–August 3, 2001 Proceedings 9, Springer, 2001, pp. 129–142.

[8] S. Ferré, M. Huchard, M. Kaytoue, S. O. Kuznetsov, A. Napoli, Formal concept analysis: from knowledge discovery to knowledge processing, A Guided Tour of Artificial Intelligence Research: Volume II: AI Algorithms (2020) 411–445.

[9] S. O. Kuznetsov, Fitting pattern structures to knowledge discovery in big data, in: Formal Concept Analysis: 11th International Conference, ICFCA 2013, Dresden, Germany, May 21-24, 2013. Proceedings 11, Springer, 2013, pp. 254–266.

[10] M. Kaytoue, S. O. Kuznetsov, A. Napoli, S. Duplessis, Mining gene expression data with pattern structures in formal concept analysis, Information Sciences 181 (2011) 1989–2001.

[11] S. O. Kuznetsov, Scalable knowledge discovery in complex data with pattern structures, in: Pattern Recognition and Machine Intelligence: 5th International Conference, PReMI 2013, Kolkata, India, December 10-14, 2013. Proceedings 5, Springer, 2013, pp. 30–39.

[12] B. Ganter, S. O. Kuznetsov, Hypotheses and version spaces, in: ICCS, volume 2746, Springer, 2003, pp. 83–95.

[13] A. Masyutin, Y. Kashnitsky, S. Kuznetsov, Lazy classification with interval pattern structures: Application to credit scoring, in: Workshop Notes, 2015, p. 43.

[14] B. Ganter, R. Wille, Formal concept analysis: mathematical foundations, Springer Science & Business Media, 2012.

[15] B. De Ville, P. Neville, Decision trees for analytics: using SAS Enterprise miner, SAS Institute Cary, NC, 2013.

Using Multimodal Clustering on Formal Contexts in Event Extraction with Neural Nets

Mikhail Bogatyrev [1], Dmitry Orlov [1], Ilya Zapyantsev [1]

[1] *Tula State University, 92 Lenin ave., Tula, Russia*

Abstract
The most of the tasks of event extraction from textual data are solved using neural networks. One of the ways to train neural networks on texts is the use of *n*-grams. In this paper, we consider *n*-grams in the form of subgraphs of conceptual graphs that have a certain semantics. Each such *n*-gram can be treated as an event. However, when using conceptual *n*-grams in neural network, their semantics are either lost or additional cumbersome structural solutions are required to preserve it. The paper proposes to use an additional information resource in the form of a hierarchy of multimodal clusters based on a multidimensional formal context, the *n*+1 dimensional tensor elements of which consist of *n*-grams and elements selected as objects. The task of event extraction is solved by the joint use of a neural network and a hierarchy of multimodal clusters. The neural network solves the classification problem and returns the class objects corresponding to the query text upon request. Then these objects are searched in multimodal clusters. These clusters contain combinations of events related to the classes of objects found by the network. This method has been studied on two data sets and demonstrates a number of advantages over known approaches to extracting events from texts using neural networks and *n*-grams.

Keywords
Event extraction, conceptual *n*-grams, Formal Concept Analysis, multimodal clustering

1. Introduction

This paper represents the ongoing research on applying Formal Concept Analysis (FCA) [1] to neural-based solutions to certain text mining tasks. Among these tasks, there is Event extraction [2].

Event extraction constitutes an area of methods for acquiring additional information in unstructured text so that it represents an answers on so-called "5W1H" questions “Who did What to Whom, Where, When and How” [2]. The answers do not necessarily have to contain all the elements of the "5W1H" question, which creates a variety of event representations.

Event extraction plays an important role in many applications in various fields. Among these applications, there are event extraction in real-time news, extract information about traffic jams from social media, extract medications in biomedical field, to name a few [2]. The event extraction task can be interpreted as a generalization of two text mining tasks: named entity recognition and relationship extraction. Accordingly, the arsenal of tools for solving these tasks is used in event extraction. Among these tools, neural networks currently play a key role.

In this paper, we investigate the use of *n*-grams extracted from text to train a neural network instead of training it on full texts. This method of learning is known and is used in a number of tasks. But we use "meaningful" conceptual *n*-grams, which contain answers to the "5W1H" questions in whole or in part. Each such *n*-gram also can be treated as an event.

However, when training a neural network on conceptual *n*-grams by standard way, the semantics of "meaningful" *n*-grams is not represented in the network. The paper proposes to use an additional

Proceedings Name, Month XX–XX, YYYY, City, Country
EMAIL: email1@mail.com (A. 1); email2@mail.com (A. 2); email3@mail.com (A. 3)
ORCID: XXXX-XXXX-XXXX-XXXX (A. 1); XXXX-XXXX-XXXX-XXXX (A. 2); XXXX-XXXX-XXXX-XXXX (A. 3)

information resource in the form of a hierarchy of multimodal clusters based on a multidimensional formal context, the n+ 1-tensor elements of which are n-grams and the elements selected as objects. The task of event extraction is solved by the joint use of a neural network and a hierarchy of multimodal clusters. The neural network solves the classification problem and returns the class objects corresponding to the query text upon request. Then these objects are searched in multimodal clusters. These clusters contain combinations of events related to the classes of objects found by the network.

Using conceptual n-grams for learning allows one to apply neural network with a simple architecture. The use of an additional resource in the form of a hierarchy of clusters built on n-grams presents additional information to obtain a solution of event extraction task.

The paper is organized as follows. In the Section 2, there are brief descriptions of event extraction task and multimodal clustering problem in FCA with mentioning the main related works in these areas. In the Section 3, the proposed approach is outlined together with its functionality. The Section 4 contains conclusion. The paper ends with acknowledgements and references sections.

2. Preliminaries and Related Work

This work is related to two principles. The first principle of *query refining* has long been known in FCA community [3]. If a query is identified with some node of conceptual model (in FCA it is conceptual lattice), then its refinements or possible answers to it are located in neighboring nodes. In our method, class objects found by the neural network act as queries. Clusters as formal concepts corresponding to queries, form a lattice in which query refining principle operates. Having fixed the cluster that best matches the query, we find the clusters adjacent to it and use this principle to extract events.

The second principle of *concept-based explanations* [4] is applied in neural networks and aims to implement the notion of a concept in the layers of deep neural networks. In the work of [5] this principle is used to build an intrinsically interpretable document classifier using a combination of FCA and approaches from applied graph theory. At the current stage of research, we do not use the semantics of conceptual n-grams directly in the neural network, limiting ourselves to the use of standard configuration networks. However, the principle of concept-based explanations is applied when working with multimodal clusters.

2.1. Neural-based event extraction from texts

The main definitions in the event extraction are the following [2]. An *event mention* usually is a phrase or sentence that describes an event in which a trigger and corresponding arguments are included. Not every sentence contains event mention, so that sentences without events must be recognized and omitted. *Event trigger* is usually a verb or a noun that most clearly expresses the core meaning of an event. We use verb-oriented algorithm for acquiring conceptual graphs from text and construct n-grams. *Event type* refers to the category to which the event corresponds. In most cases, event types are predefined manually, categorized by event triggers. For instance, we may be interested on existing some verbs in the text, i.e. attack, shoot, etc. *Event arguments* are the main attributes of events. They are usually entity mentions describing the event state change, involving who, what, when, where, and how. *An argument role* is a function or position that an event argument performs in the relationship between the event argument and the trigger.

There are two directions in event extraction. They are *Closed-domain event extraction* and *Open-domain event extraction*. *Closed-domain event extraction* uses predefined event schema to discover and extract desired events of particular type from text. An event schema contains several event types and their corresponding event structures. *Open-domain event extraction* aims at detecting events from texts and in most cases, also clustering similar events via extracted event keywords. Event keywords refer to those words/phrases mostly describing an event, and sometimes keywords are further divided into triggers and arguments.

There are a large number of works devoted to event extraction from the text. The review [2] contains 245 references. Among the works closest to our topic and using neural networks, we highlight the following two papers.

In the work of [6] the Abstract Meaning Representation (AMR) graphs acquired from text are used. AMR graph is a rooted directed acyclic graph where the nodes represent concepts and the edges represent relations between these concepts. Events are considered as subgraphs of AMR graph. Neural network is used to identify an event subgraphs. The neural network is trained on a tagged textual corpus. Tagging includes the definition of events and their arguments. Methodology of event extraction presented in the work of [6] needs external information resources. Since this work is about biomedical event extraction, these resources are knowledge base containing relations between proteins and large corpus of unannotated text that contain protein mentions.

The work of [7] represents recent results of nested event extraction from biomedical domain data. In this work, event extraction model named as DeepEventMine is proposed. It extracts multiple overlapping directed acyclic graph structures from a sentence. An event is called nested event when it has other events in its arguments, while an event is called flat event when there are only entities in its arguments. According to DeepEventMine model, an event consists of a trigger and zero or more arguments. A trigger is a textual mention that denotes the presence of an event in text. Triggers, events, arguments and roles are all united in graph structures. The model is strongly tied to pre-trained networks, primarily to the BERT network. The implementation of the model requires large computing resources.

Based on the mentioned works and other works in this research area, the following conclusions can be drawn.

1. Graph models of event representation are used in the works.
2. Neural networks used in event extraction have a complex architecture.
3. To extract events, external resources are needed in the form of text corpora, thesauri, and databases.

2.2. Multimodal clustering in FCA

In FCA, multimodal clustering is formulated as follows.

If $R \subseteq D_1 \times D_2 \times ... \times D_n$ is a relation on data domains $D_1, D_2, ..., D_n$ then *formal context* is an n +1 set:

$$\mathbb{K} = <K_1, K_2, ..., K_n, R> \quad (1)$$

where $K_i \subseteq D_i$. *Multimodal clusters* on the context (1) are n – sets

$$\mathbb{C} = < X_1, X_2, ..., X_n > \quad (2)$$

which have the closure property [8]:

$$\forall u = (x_1, x_2, ..., x_n) \in X_1 \times X_2 \times, ..., \times X_n, u \in R \quad (3)$$

and $\forall j = 1, 2, ..., n, \forall x_j \in D_j \setminus X_j < X_1, ..., X_j \cup \{x_j\}, ..., X_n >$ does not satisfy (3).

A multimodal cluster is a subset in the form of combinations of elements from different sets K_i. It is also defined as a closed n-set [9] since the closure property (3) provides its "self-sufficiency": it cannot be enlarged without violating (2).

Formal concepts on multimodal formal context are those multimodal clusters where *for all* $u = (x_1, x_2, ..., x_k) \in X_1, X_2, ..., X_k, u \in R$ and k is maximally possible. In other words, they are the largest possible k-dimensional hypercubes completely filled with units. The concept of the density of a multimodal cluster is introduced in FCA and formal concepts are interpreted as absolutely dense clusters [9].

3. Methodology

The considered approach is implemented by performing the following steps.

1. Conceptual *n*-grams are built on the text under processing. We consider every such *n*-gram as a graph in the form of a tree. For a given sentence, it may be dependency parse tree, AMR-graph, conceptual graph or any other *n*-gram model. Depending on the type of *n*-grams, their construction can be performed by an appropriate software tool. All the currently most well known variants of *n*-grams are supported by parser programs. It is necessary to ensure the storage of *n*-grams, for example, in a database. Some text corpora store *n*-grams as tagging.

2. A multidimensional formal context is formed as an *n*+1-dimensional tensor. Every point of this tensor contains an object and a *n*-gram it corresponding. Objects can be documents containing texts, terms, entity names, text topics, etc. Multimodal clusters are built on this formal context.

3. The task of event extraction is solved by the joint use of a neural network and a hierarchy of multimodal clusters. The neural network solves the classification problem and, upon request, returns objects-classes corresponding to the query text. Then the clusters corresponding to these objects are determined. They contain combinations of events related to the query text.

3.1. Experiments and Early Results

Consider implementation of the outlined methodology and its experimental study.

Data sets. We explored our methodology on two datasets. The first dataset is The Stanford Question Answering Dataset (SQuAD) [10] . In this dataset there are topics, corresponding texts, questions and answers. This dataset is often used to train neural networks applied in question-answering systems. Extracting events in SQuAD corresponding to the semantic template “who, what, with whom and where and when” will allow one to form answers to questions related to the template elements.

The second dataset (KaggleTDC) is a collection of approximately 1000 newsgroup documents from 10 different newsgroups [11]. This dataset differs from SQuAD in that its newsgroups more overlap by words. Therefore, we expected greater uncertainty in determining the newsgroup from the text by the neural network.

Conceptual *n*-grams. The sequence of *n* tokens in the text is called an *n*-gram. The token can be a phoneme, a syllable, a letter, a word or base pairs according to the application. We construct *n*-grams by acquiring conceptual graphs [12] from text and selecting their subgraphs having three, four or five concepts connected by *agent*, *patient* or *attribute* relations. Accordingly, we have trigrams, 4-grams and 5-grams.

Figure 1 shows an example of conceptual graphs corresponding to the sentence "*Dioxins are highly toxic, and health effects on humans include reproductive, developmental, immune and hormonal problems*" related to the topic of the Immune system.

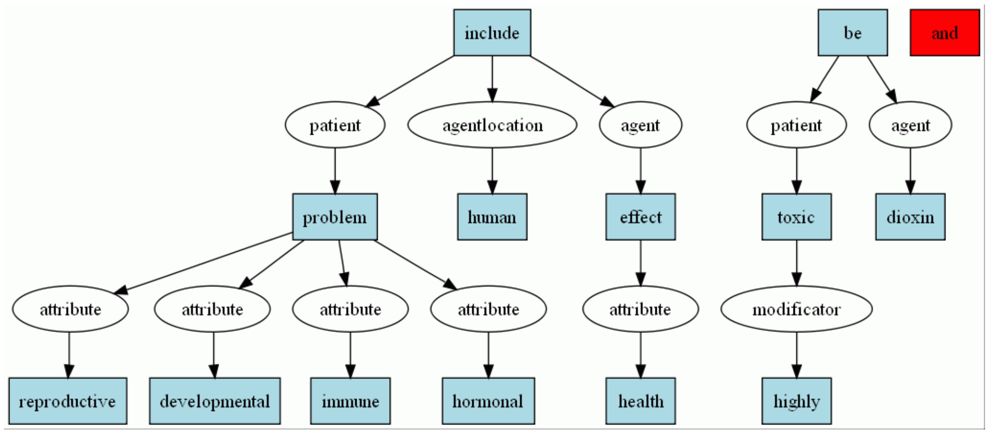


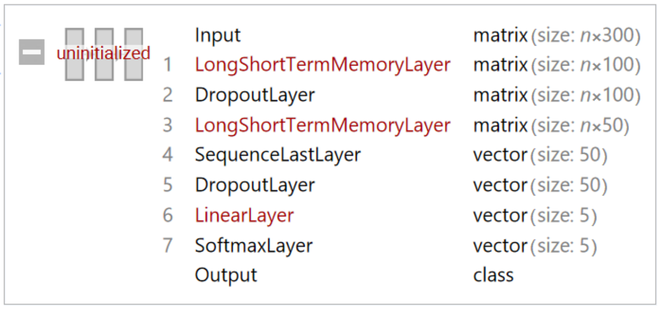
Figure 1: An example of conceptual graphs corresponding to the sentence "Dioxins are highly toxic, and health effects on humans include reproductive, developmental, immune and hormonal problems"

In Figure 1, two disconnected conceptual graphs correspond to the sentence above. By selecting the *agent* and *patient* relationships in graphs, we have the trigrams “effect – include - problem” and “dioxin

– be - toxic". When adding *attribute* relations, we get five 5-grams based on the first trigram with the concepts of *health, hormonal, immune, developmental, reproductive*. For example, it will be a 5-gram "health – effect - include – problem – hormonal", which corresponds to the event template given above. All these conceptual *n*-grams may be considered as events.

Conceptual graphs used in our work constitute a special case of AMR graphs. Conceptual graphs represent the semantics of individual sentences by acquiring from text known semantic roles as *agent*, *patient*, *attribute*, etc. We do not use external information resources and train neural network on n-grams extracted from conceptual graphs.

Neural Network. We use a compact chain recurrent neural network of standard architecture. An example of a structure of such network for 5 classes is shown in Figure 2.



	Input	matrix (size: n×300)
1	LongShortTermMemoryLayer	matrix (size: n×100)
2	DropoutLayer	matrix (size: n×100)
3	LongShortTermMemoryLayer	matrix (size: n×50)
4	SequenceLastLayer	vector (size: 50)
5	DropoutLayer	vector (size: 50)
6	LinearLayer	vector (size: 5)
7	SoftmaxLayer	vector (size: 5)
	Output	class

Figure 2: An example of a structure of neural network for 5 classes.

The network is trained on *n*-grams generated from the texts of the dataset. When training the network, several optimization methods were used: ADAM, RMSProp, SGD, Signed.

The input data in the network are phrases in natural language, usually in the form of a question. Conceptual *n*-grams are constructed corresponding to input phrases. If no *n*-grams correspond to a phrase, the first *n* words of the phrase come to the input layer. If several *n*-grams correspond to a phrase, then they consistently arrive at the input layer. The output of the network is the object-class corresponding to the text query. Training and network operation using conceptual *n*-grams were compared with the training method in which *n*-grams formed by using a sliding context window (SCW). The results are presented in the tables 1, 2 and on Figure 3.

The tables 1 and 2 contain some results of training and network operations for conceptual *n*-grams and for SCW *n*-grams constructed on the SQuAD dataset and on the KaggleTDC dataset.

Table 1
Results of training and network operations for SQuAD dataset

Parameter	Conceptual *n*-grams	SCW *n*-grams
Number of objects	636	30166
Time training (sec.)	**0.2854565**	5.2802665
Accuracy	**0.796875**	0.728207

Table 2
Results of training and network operations for KaggleTDC dataset

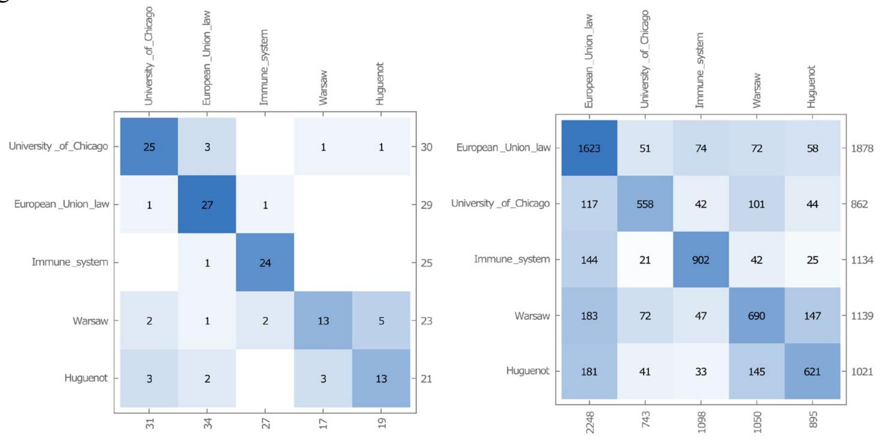
Parameter	Conceptual *n*-grams	SCW *n*-grams
Number of objects	1339	67076
Time training (sec.)	**2.97245**	163.408
Accuracy	0.802	**0.811**

The table 3 contains information about the five most voluminous topics-classes from the SQuAD data set.

Table 3
F1-Score for five most voluminous topics-classes from the SQuAD data set

Class	Conceptual *n*-grams	SCW *n*-grams
European_Union_law	**0.85714**	0.78671
University_of_Chicago	**0.81967**	0.69532
Immune_system	**0.92307**	0.80824
Warsaw	**0.65**	0.63042
Huguenot	**0.65**	0.64822

Figure 3 shows confusion matrices for neural network learned by conceptual n-grams and by SCW *n*-grams.

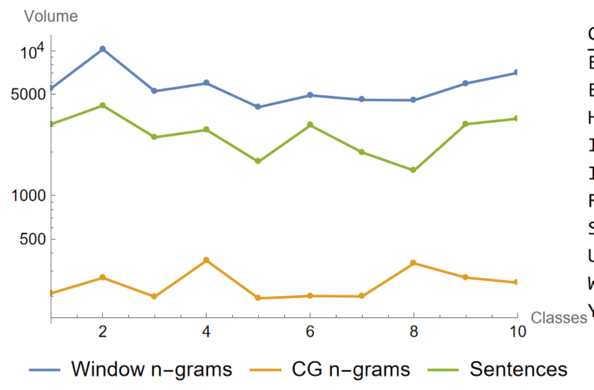


(a) (b)

Figure 3: Confusion matrices for neural network learned by conceptual *n*-grams (a) and by SCW n-grams

In general, from the most general positions, it can be argued that the more words are used when training a neural network, the better it works. The latter means, among other things, that the network responds to requests more meaningfully. Here we do not take into account the effect of overfitting.

Among others, there are two ways to train neural networks on textual data. The first one uses a bag of words – all the words of the text without stop words. The second uses *n*-grams obtained by applying a sliding context window. Both methods use a lot of words, but they clearly do not reflect the semantics of the text, since they are simple sequences of words. Figure 4 shows how the volumes of *n*-grams and sentences of the text are correlated.



Class name	Number
Economic_inequality	1
European_Union_law	2
Huguenot	3
Immune_system	4
Islamism	5
Rhine	6
Scottish_Parliament	7
University_of_Chicago	8
Warsaw	9
Yuan_dynasty	10

Figure 4: *N*-gram volumes for different classes of the SQuAD dataset.

Conceptual *n*-grams that we form consist of a small number of words and their number is more than 10 times less than the number of *n*-grams formed by using SCW as can be seen in Figure 4. Based on this, it can be assumed that the use of conceptual *n*-grams will reduce the performance of the neural network. In reality, this does not happen, as can be seen from Table 2, Table 3 and Figure 3. In our opinion, this is due precisely to the presence of semantics in conceptual *n*-grams.

As the value of *n* in conceptual *n*-grams increases, the quality of the network increases too, as can be seen from Table 4.

Table 4
Variants of *n*-grams and results of their application

Type of *n*-gram	*F1*-score
3-gram	0,704036
4-gram with agent role	0,67713
4-gram with patient role	0,737643
5-gram	0,817518
Complex 3-5-gram	0,8998

Joint use of neural net and multimodal clusters. The neural network is trained on *n*-grams, which are interpreted as events. The texts at the network input are also converted into n-grams. As a result, the network works only with *n*-grams. When solving a classification task, in response to an input query, the network returns a set of class names for the SQuAD dataset and a set of newsgroup names for the KaggleTDC dataset. It was found in experiments that in these sets there are often objects-classes with similar probabilities, which makes it difficult to choose between them. For these purposes, multimodal clusters are used. There are such clusters-formal concepts, where subsets of objects include objects-classes with similar probabilities. In such clusters, all n-grams are combined with all objects. As a result, we get a set of events corresponding to the request to the neural network.

Multimodal clustering is performed using an evolutionary algorithm that allows obtaining Pareto-optimal solutions for several criteria of cluster optimality. The framework [13] is used for multimodal clustering. The problem of multimodal clustering is formulated as a problem of multiobjective optimization. In the experiments, two clustering optimality criteria were used, the cluster volume and its density. These criteria contradict each other, so compromise Pareto-optimal solutions are needed. First of all, we were interested in absolutely dense clusters being formal concepts. In them, all objects are combined with all elements of *n*-grams. Large and dense clusters are interesting because combinations of elements of its subsets set a property that manifests itself on a large number of elements and, possibly, means a regularity. However, often the clustered data is sparse and the existence of large and dense clusters on them is unlikely. Therefore, when selecting clusters, a trade-off between density and volume is provided by the algorithm.

The quality characteristics that are usually used in the classification problem (accuracy and *F1*-score) are integral and do not reflect the quality of individual solutions, their semantics. In the task of event extraction, it is important to detect events as facts and present the composition of events. In our experiments, we still detect events as facts. In multimodal clusters, we count the number of *n*-grams and fix their composition. Table 5 shows the content of several multimodal clusters containing pairs of class objects. For every pair there are corresponding number of *n*-grams shown in the table.

Table 5
Contents of some multimodal clusters

Class objects	The number of *n*-grams
University of Chicago Warsaw	8
Immune_system European_Union_law	2
European_Union_law Scottish Parliament	28

In the Table 5, *University of Chicago* and *Warsaw* classes are linked by eight *n*-grams having mutual words in corresponding texts. *Immune_system* and *European_Union_law* classes are practically unlinked since their mutual words in two *n*-grams are commonly used. It is evident that *European_Union_law* and *Scottish Parliament* classes are closely linked by 28 *n*-grams.

As a result, in clusters we find events that have common class objects and class objects that have common events.

Obviously, in the task of event extraction, only *n*-grams are not enough for the end user and additional information is needed. The *n*-grams found in multimodal clusters correspond to text fragments that can be presented to the user in response to a request. In this case, events take the form of descriptions understandable to the user. Such a solution may be designed in the form of a user interface focused on a specific subject area.

4. Conclusion

In this paper, an approach to training neural networks on *n*-grams in the form of conceptual graphs is proposed, which allows using a recurrent network with a simple architecture. Another contribution is the use of multimodal clusters as an additional information resource in the task of event extraction. The results of the application of these solutions are the following.

1. The neural network learns on *n*-grams no worse than in words, but it works faster.
2. The use of an additional resource in the form of a multidimensional formal context and multimodal clusters makes it possible to improve the interpretability of the results of the neural network.

In the future, an analytical explanation of the experimental results obtained in this work is necessary.

5. Acknowledgements

The reported study was funded by Russian Foundation of Basic Research, the research project № 19-07-01178 and RFBR and Tula Region research project № 19-47-710007.

6. References

[1] Ganter, Bernhard; Stumme, Gerd; Wille, Rudolf, eds., Formal Concept Analysis: Foundations and Applications, Lecture Notes in Artificial Intelligence, No. 3626. (2005). Springer-Verlag. Berlin

[2] W. Xiang and B. Wang, "A Survey of Event Extraction from Text." IEEE Access, vol. 7, pp. 173111-173137, 2019, doi: 10.1109/ACCESS.2019.2956831.

[3] Carpineto, Claudio, Romano, Giovanni. A Lattice Conceptual Clustering System and Its Application to Browsing Retrieval. Machine Learning. 24. 95-122. (1996). doi: 10.1007/BF00058654.

[4] D. Chen, A. Fisch, J. Weston, A. Bordes, Reading Wikipedia to answer open-domain questions, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 1870–1879. doi:10.18653/v1/P17- 1171.

[5] Parakal E. G., Kuznetsov S. Intrinsically Interpretable Document Classification via Concept Lattices, in: Proceedings of the 10th International Workshop "What can FCA do for Artificial Intelligence?" / Ed. by S. Kuznetsov, A. Napoli, S. Rudolph. Vol. 3233. CEUR Workshop Proceedings, 2022. Ch. 2. P. 9-22.

[6] Sudha Rao, Daniel Marcu, Kevin Knight, Hal Daum´e III. Biomedical Event Extraction using Abstract Meaning Representation. Proc. of the BioNLP 2017 workshop, Canada, August 4, 2017.

[7] Ananiadou, S. DeepEventMine: End-to-end Neural Nested Event Extraction from Biomedical Texts. Bioinformatics (2020). https://doi.org/10.1093/bioinformatics/btaa540

[8] Cerf, L., Besson, J., Robardet, C., Boulicaut, J.F.: Closed Patterns Meet N-ary Relations. In: ACM Trans. Knowl. Discov. Data. 3, 1, Article 3, (2009). 36 p.

[9] Ignatov D. I., Gnatyshak D. V., Kuznetsov, S. O., Mirkin, B., G.: Triadic Formal Concept Analysis and triclustering: searching for optimal patterns. Mach. Learn. 101:271–302. (2015).
[10] The Stanford Question Answering Dataset. URL: https://rajpurkar.github.io/SQuAD-explorer/
[11] The Kaggle Dataset Text Document Classification. URL: https://www.kaggle.com/datasets/jensenbaxter/10dataset-text-document-classification.
[12] Sowa J.: Conceptual Graphs: Draft Proposed American National Standard, International Conference on Conceptual Structures ICCS-99, Springer (1999).
[13] Mikhail Bogatyrev, Dmitry Orlov. Framework for Pareto-Optimal Multimodal Clustering. In Sergei O. Kuznetsov, Amedeo Napoli, Sebastian Rudolph (Eds.): The 10th International Workshop "What can FCA do for Artificial Intelligence?" FCA4AI 2022, co-located with IJCAI-ECAI 2022, July 23 2022, Vienna, Austria. CEUR Workshop Proceedings. Vol. 3233. Pp. 51–62.

Full polynomial probabilistic FCA-based knowledge extraction

Dmitry V. Vinogradov[1,*,†]

[1]*Dorodnicyn Computing Center, Federal Research Center "Computer Science and Control", Russian Academy of Sciences, 40 Vavilova St., Moscow, 119333, Russian Federation*

Abstract

The article demonstrates computational efficiency of the probabilistic approach to knowledge extraction using the FCA. In addition to the result previously proved by the author on sufficiency of a polynomial number of hypotheses (concepts) about the causes of the target property under study, this paper will give a polynomial upper bound on the average running time of the algorithm for generating one concept. The proven result concerns a family of algorithms based on coupling Markov chains for arbitrary formal contexts formed from the positive part of training sets. To get a good estimate for the length of trajectory (before entering to some ergodic state) of such a chain, we had to enrich the representation of the training sample by adding negation for every original binary attribute.

Keywords

formal concept, coupling Markov chain, mean length of trajectory, computational complexity

1. Introduction

The extraction of knowledge using a binary similarity operation began in the early 1980s in the works of Prof. V.K. Finn, who proposed the JSM-method of automatic generation of hypotheses [1, 2].

This approach was named after the English philosopher, economist and logician John Stuart Mill, whose ideas on Inductive Logic [3] served as the starting point of the JSM-method. The key component of this approach is a binary similarity operation. In the beginning, this operation was considered in isolation: most often as the intersection of sets of binary attributes describing training examples. In this case, it was a way of finding a set of common attributes. Initially, domains of objects (training and test examples) were Boolean algebras.

Then S.O. Kuznetsov proposed [4] to apply Formal Concept Analysis [5] to JSM-paradigm. This discovery led to extension of domains of application by those, that can be described by general lattices, and to invention of more efficient algorithms [6].

However, the JSM-method has a number of significant limitations that do not allow it to cope with training samples of moderate size. One of them is exponential explosion, when a small training context generates exponentially large number of concepts [7]. Another one is the

11-th Workshop 'What Can FCA Do for Artificial Intelligence?' co-located with IJCAI 2023, August 20, 2023, Macao, S.A.R.

vinogradov.d.w@gmail.com (D. V. Vinogradov)
http://www.ccas.ru (D. V. Vinogradov)
0000-0001-5761-4706 (D. V. Vinogradov)

appearance of so-called 'phantom' concepts as accidental similarities between small number of training objects each of which belongs to a different concept with larger extent [8]. It can be argued that the appearance of such hypotheses corresponds to the over-fitting phenomenon. This statement was experimentally confirmed in the master thesis of L.A. Yakimova [9].

To overcome these limitations, the author [10] proposed to use a probabilistic approach. The idea is to generate a random sample of concepts by trajectories of Markov chain making random walks through the concept lattice. We named our approach the VKF method in honor of V.K. Finn and because of the abbreviation of the Russian term "Probabilistic Combinatorial Formal method" to indicate effective processing using probabilistic algorithms and FCA of various combinations of training objects for generating concepts.

Such algorithms are based on the "Close-by-One" operations CbO, for which the part with respect to objects was proposed earlier by S.O. Kuznetsov [11] in the eponymous CbO algorithm for exhaustive generation of all candidates for hypotheses, the number of which in some cases may be exponentially large. Using these operations, the author proposed to generate a polynomial-size random subset of concepts, each element of which corresponds to one trajectory of random walk across the corresponding lattice.

The author [12] has proved that it is sufficient to generate $\frac{n \cdot \ln 2 - \ln \delta}{\varepsilon}$ random concepts in order to correctly predict all the ε-important test objects with the reliability of $1 - \delta$.

Therefore, the single obstacle for polynomial complexity of full scheme of the VKF-method is the absence of polynomial upper bound on the length of trajectories of Markov chain. The main result of this paper is polynomial upper bound on the average length of trajectories of the coupling Markov chain when the training context is dichotomized, i.e., expanded by additional binary attributes that correspond to negations of all original attributes. Such expansion is useful if the absence of an original attribute is allowed to be a part of cause for the target attribute. Previously, only special cases of formal contexts (for instance, Boolean algebra and linear order) were investigated. The new result concerns the general case of arbitrary lattice.

2. Background

2.1. Basic definitions and facts of FCA

Here we recall some basic definitions and facts from Formal Concept Analysis (FCA) [5].

A **(formal) context** is a triple (G, M, I) where G and M are finite sets and $I \subseteq G \times M$. The elements of G and M are called **objects** and **attributes**, respectively. As usual, we write gIm instead of $\langle g, m \rangle \in I$ to denote that object g has attribute m.

For $A \subseteq G$ and $B \subseteq M$, define

$$A' = \{m \in M : \forall g \in A(gIm)\}, \tag{1}$$

$$B' = \{g \in G : \forall m \in B(gIm)\}; \tag{2}$$

so A' is the set of attributes common to all the objects in A and B' is the set of objects possessing all the attributes in B. The maps $(\cdot)' : A \mapsto A'$ and $(\cdot)' : B \mapsto B'$ are called **derivation operators** (also **polars**) of the context (G, M, I).

A **concept** of the context (G, M, I) is defined to be a pair (A, B), where $A \subseteq G$, $B \subseteq M$, $A' = B$, and $B' = A$. The first component A of the concept (A, B) is called the **extent** of the concept, and the second component B is called its **intent**. The set of all concepts of the context (G, M, I) is denoted by $\mathbf{B}(G, M, I)$.

Let (G, M, I) be a context. For concepts (A, B) and (C, D) in $\mathbf{B}(G, M, I)$ we write $(A, B) \leq (C, D)$, if $A \subseteq C$. The relation $\leq$ is a **partial order** on $\mathbf{B}(G, M, I)$.

A subset $A \subseteq G$ is the extent of some concept if and only if $A'' = A$ in which case the unique concept of which A is the extent is (A, A'). Similarly, a subset B of M is the intent of some concept if and only if $B'' = B$ and then the unique concept with intent B is (B', B).

Proposition 1. *Let (G, M, I) be a context. Then $(\boldsymbol{B}(G, M, I), \leq)$ is a lattice with join and meet given by*

$$\bigvee_{j \in J} (A_j, B_j) = ((\bigcup_{j \in J} A_j)'', \bigcap_{j \in J} B_j), \tag{3}$$

$$\bigwedge_{j \in J} (A_j, B_j) = (\bigcap_{j \in J} A_j, (\bigcup_{j \in J} B_j)''); \tag{4}$$

Corollary 1. *For context (G, M, I) the lattice $(\boldsymbol{B}(G, M, I), \leq)$ has (M', M) as the bottom element and (G, G') as the top element. In other words, for all $(A, B) \in \boldsymbol{B}(G, M, I)$ the following inequalities hold:*

$$(M', M) \leq (A, B) \leq (G, G'). \tag{5}$$

For $(A, B) \in \mathbf{B}(G, M, I)$, $g \in G$, and $m \in M$ define

$$CbO((A, B), g) = (A, B) \vee (\{g\}'', \{g\}'), \tag{6}$$

$$CbO((A, B), m) = (A, B) \wedge (\{m\}', \{m\}''), \tag{7}$$

so according to (4) $CbO((A, B), g)$ is equal to $((A \cup \{g\})'', B \cap \{g\}')$ and according to (3) $CbO((A, B), m)$ is equal to $(A \cap \{m\}', (B \cup \{m\})'')$.

The useful properties of introduced operations are summarized in the following Lemmas.

Lemma 1. *Let (G, M, I) be a context, $(A, B) \in \boldsymbol{B}(G, M, I)$, $g \in G$, and $m \in M$. Then*

$$g \in A \Rightarrow CbO((A, B), g) = (A, B), \tag{8}$$

$$m \in B \Rightarrow CbO((A, B), m) = (A, B), \tag{9}$$

$$g \notin A \Rightarrow (A, B) < CbO((A, B), g), \tag{10}$$

$$m \notin B \Rightarrow CbO((A, B), m) < (A, B). \tag{11}$$

Lemma 2. *Let (G, M, I) be a context, $(A, B), (C, D) \in \boldsymbol{B}(G, M, I)$, $g \in G$, and $m \in M$. Then*

$$(A, B) \leq (C, D) \Rightarrow CbO((A, B), g) \leq CbO((C, D), g), \tag{12}$$

$$(A, B) \leq (C, D) \Rightarrow CbO((A, B), m) \leq CbO((C, D), m). \tag{13}$$

2.2. Random walks by coupled Markov chain

To avoid the open problem of calculation of mixing time of general Markov chain we proposed [10] to use the coupled Markov chain for random walks across the concept lattice. The states of this chain are ordered pairs of concepts. The stopping time of the random walk algorithm is the first moment of entering to some ergodic (recurrent) state of the coupled Markov chain. Every ergodic state of the coupled Markov chain is a pair of equal concepts. Denote the set of such states by E.

```
**Data:** context $(G, M, I)$, external function $CbO( \, , \, )$
**Result:** random concept $(A, B) \in \mathbf{B}(G, M, I)$
$X := G \sqcup M$; $(A, B) := (M', M)$; $(C, D) = (G, G')$;
**while** $((A \neq C) \vee (B \neq D))$ **do**
    select random element $x \in X$;
    $(A, B) := CbO((A, B), x)$;
    $(C, D) := CbO((C, D), x)$;
**end**
```

Algorithm 1: Coupling Markov chain

The algorithm terminates when the upper and lower concepts coincide. The condition on remaining of ordering between two concepts $(A, B) \leq (C, D)$ at any intermediate step of the while loop of Algorithm 1 follows from Lemma 2.

The classical theorem of Markov chain Theory about transient (non-ergodic) states [13] implies almost surely termination of algorithms 1, i.e. finiteness of a trajectory until it enters to some ergodic state with probability 1.

Consider the moment $T_i(E) = \min\{t : X_t \in E, X_0 = s_i\}$ of the first entering to E, starting with an arbitrary transient state $s_i = (\langle A, B\rangle < \langle C, D\rangle) \notin E$.

Theorem 1. *The moment $T_i(E)$ is Markov one for every transient state s_i.*

Proof. We need to prove $\mathbb{P}\left[T_i(E) < \infty \mid X_0 = s_i\right] = 1$.

Use decomposition $\{X_t \in E, X_0 = s_i\} = \bigcup_{n \leq t} U_n(s_i)$, where

$$U_n(s_i) = \{X_n \in E, X_{n-1} \notin E, \dots, X_1 \notin E, X_0 = s_i\}.$$

Transient States Theorem asserts

$$\lim_{t \to \infty} \mathbb{P}\left[X_t \notin E \mid X_0 = s_i\right] \to 0. \tag{14}$$

Disjointedness of different $U_n(s_i)$ and formula (14) imply

$$\mathbb{P}\{X_t \in E \mid X_0 = s_i\} = \sum_{n \leq t} \mathbb{P}\left[U_n(s_i) \mid X_0 = s_i\right] \to 1,$$

if $t \to \infty$.

Since $U_n(s_i) = \{T_i(E) = n\}$ the σ-additivity leads to needed conclusion. □

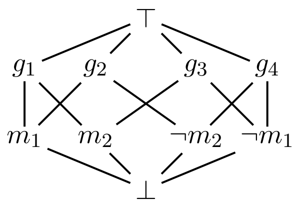
As direct corollary of the theorem 1 we conclude that the termination of algorithm 1 takes place almost surely (i.e. with probability 1).

The goal of current research is to obtain a polynomial upper bound on the average length of trajectories of the coupling Markov chain. In general, it is an open problem. In sequel, we'll provide such bound, when the training context is expanded by additional binary attributes that correspond to negations of all existing attributes (dichotomic expansion).

Example 1. *Dichotomic expansion of the left context is the right one.*

$G \times M$	m_1	m_2		$G \times M^+$	m_1	$\neg m_1$	m_2	$\neg m_2$
g_1	*1*	*1*		g_1	*1*	*0*	*1*	*0*
g_2	*1*	*0*		g_2	*1*	*0*	*0*	*1*
g_3	*0*	*1*		g_3	*0*	*1*	*1*	*0*
g_4	*0*	*0*		g_4	*0*	*1*	*0*	*1*

The expanded context corresponds to the lattice



where $\top = \langle \emptyset, \{m_1, \neg m_1, m_2, \neg m_2, \} \rangle$, $g_j = \langle \{g_j\}, \{g_j\}' \rangle$, $m_j = \langle \{m_j\}', \{m_j\} \rangle$, $\neg m_j = \langle \{\neg m_j\}', \{\neg m_j\} \rangle$, *and* $\bot = \langle \{g_1, g_2, g_3, g_4\}, \emptyset \rangle$.

The coupled Markov chain starts with state $(\bot \leq \top)$. *A trajectory of the random walk depends on random choices from* $G \sqcup M^+$.

Consider an example of such trajectory. Assume that g_1 *is selected at the 1st step, then the chain goes to state* $(\bot \leq g_1)$. *The choice of* g_2 *at the 2nd step leads to* $(\bot \leq m_1)$. *If the chain selects* $\neg m_1$ *at the 3rd step, then the state becomes* $(\neg m_1 \leq \top)$. *The choice of* g_4 *at the 4th step leads to* $(\neg m_1 \leq g_4)$. *After selection of* g_3 *at the 5th step the trajectory goes to ergodic state* $(\neg m_1 \leq \neg m_1)$, *and algorithm 1 stops.*

3. Technical Tools

In [14] the author developed a useful tool to estimate the average length of trajectories of coupling Markov chain through recurrence relations.

Lemma 3.

$$\mathbb{E}\left[T_i(E)\right] = 1 + \sum_{s_j \notin E} \mathbb{E}\left[T_j(E)\right] \cdot \mathbb{P}\left[X_1 = s_j | X_0 = s_i\right]$$

for every $s_i \notin E$.

Proof. Additivity of the average gives

$$\mathbb{E}\left[T_i(E)\right] = \sum_{n=1}^{\infty} n \cdot \mathbb{P}\left[U_n(s_i) | X_0 = s_i\right], \tag{15}$$

where $U_n(s_i) = \{X_n \in E, X_{n-1} \notin E, \ldots, X_1 \notin E, X_0 = s_i\}$.

Then

$$\mathbb{E}\left[T_i(E)\right] = \sum_{n=1}^{\infty} n \cdot \mathbb{P}\left[U_n(s_i)|X_0 = s_i\right] =$$
$$= \sum_{n=1}^{\infty} \mathbb{P}\left[U_n(s_i)|X_0 = s_i\right] + \sum_{n=2}^{\infty} (n-1) \cdot \mathbb{P}\left[U_n(s_i)|X_0 = s_i\right] =$$
$$= 1 + \sum_{k=1}^{\infty} k \cdot \mathbb{P}\left[X_{k+1} \in E, X_k \notin E, \ldots, X_1 \notin E | X_0 = s_i\right] = 1+$$
$$+ \sum_{s_j \notin E} \sum_{k=1}^{\infty} k \cdot \mathbb{P}\left[X_{k+1} \in E, X_k \notin E, \ldots, X_2 \notin E | X_1 = s_j\right] \cdot \mathbb{P}\left[X_1 = s_j | X_0 = s_i\right] =$$
$$= 1 + \sum_{s_j \notin E} \mathbb{E}\left[T_j(E)\right] \cdot \mathbb{P}\left[X_1 = s_j | X_0 = s_i\right].$$

Here we sequentially use identity (15), the Markov property for moment $T_i(E)$ (theorem 1) and the Law of Total Probability. □

This easily results in an upper bound of the order $O(n \cdot \ln n)$ on the average length of trajectories of algorithm 1 for n-dimensional Boolean algebra case.

A more striking result from [14] concerns the average trajectory length of the algorithm 1 for linear orders. Here the upper bound of 4 on the average length does not depend on the number of elements of the linear order.

Example 2. *Apply lemma 3 to the lattice from example 1.*

This lattice allows us to define a distance between ordered candidates (i.e. components of a state). State $s_0 = (\bot \leq \top)$ has distance 3. States $s_1 = (\bot \leq g_1), \ldots, s_4 = (\bot \leq g_4)$, $s_5 = (m_1 \leq \top)$, $s_6 = (m_2 \leq \top)$, $s_7 = (\neg m_2 \leq \top)$, $s_8 = (\neg m_1 \leq \top)$ have distance 2. States with distance 1 are divided into 2 groups (external and internal ones). External states are $s_9 = (\bot \leq m_1)$, $s_{10} = (\bot \leq m_2)$, $s_{11} = (\bot \leq \neg m_2)$, $s_{12} = (\bot \leq \neg m_1)$, and $s_{13} = (g_1 \leq \top), \ldots, s_{16} = (g_4 \leq \top)$. Internal states correspond to edges $s_{17} = (m_1 \leq g_1), \ldots, s_{24} = (\neg m_1 \leq g_4)$. The rest states are ergodic ones (with distance 0).

Denote the length of trajectory starting from state s_0 by T_3. The states with distance 2 determine a random walk of length T_2. External states initiate trajectories of length T_1^E, and internal ones start trajectories of length T_1^M.

Lemma 3 leads to $\mathbb{E}T_3 = 1 + \mathbb{E}T_2$ and system of equations

$$\begin{cases} \mathbb{E}T_2 = 1 + \frac{3}{8}\mathbb{E}T_2 + \frac{2}{8}\mathbb{E}T_1^E + \frac{2}{8}\mathbb{E}T_1^M \\ \mathbb{E}T_1^E = 1 + \frac{1}{8}\mathbb{E}T_2 + \frac{2}{8}\mathbb{E}T_1^E + \frac{2}{8}\mathbb{E}T_1^M \\ \mathbb{E}T_1^M = 1 + \frac{2}{8}\mathbb{E}T_1^E + \frac{2}{8}\mathbb{E}T_1^M \end{cases} \tag{16}$$

The solution $\mathbb{E}T_2 = \frac{288}{72} = 4$ of the system (16) leads to the value $\mathbb{E}T_3 = 1 + 4 = 5$ of the average length of trajectory of algorithm 1 for the context considered in example 1.

Now we extend component-wise the CbO operations to states of coupling Markov chain

$$CbO((\langle A,B\rangle \leq \langle C,D\rangle),g) = (CbO(\langle A,B\rangle,g) \leq CbO(\langle C,D\rangle,g))$$

and

$$CbO((\langle A,B\rangle \leq \langle C,D\rangle),m) = (CbO(\langle A,B\rangle,m) \leq CbO(\langle C,D\rangle,m)).$$

Then we define a (partial) order between states $s_i = (\langle A_i,B_i\rangle \leq \langle C_i,D_i\rangle)$ and $s_j = (\langle A_j,B_j\rangle \leq \langle C_j,D_j\rangle)$ of coupling Markov chain as following

$$s_j \leqslant s_i \Leftrightarrow \langle A_i,B_i\rangle \leq \langle A_j,B_j\rangle \leq \langle C_j,D_j\rangle \leq \langle C_i,D_i\rangle. \tag{17}$$

Lemma 2 easily implies

Lemma 4. *For any ordered pair of states $s_j \leqslant s_i$, any $g \in G$, and any $m \in M$ $CbO(s_j,g) \leqslant CbO(s_i,g)$ and $CbO(s_j,m) \leqslant CbO(s_i,m)$ hold.*

We denote the number of training objects by $k = |G|$ and the number of attributes by $n = |M|$.

Lemma 5. *$\mathbb{E}T_j(E) \leq \mathbb{E}T_i(E)$ for any ordered pair of transient states $s_j \leqslant s_i$ of coupling Markov chain.*

Proof. Define coupled random walk of ordered pair of states $X_t \leqslant Y_t$ as following:

$$\mathbb{P}\left[X_1 = s'_j, Y_1 = s'_i \mid X_0 = s_j, Y_0 = s_i\right] =$$
$$= \begin{cases} \frac{l}{n+k}, & l = |\{g \in G : s'_j = CbO(s_j,g), s'_i = CbO(s_i,g)\}| + \\ & +|\{m \in M : s'_j = CbO(s_j,m), s'_i = CbO(s_i,m)\}| \\ 0, & \neg\exists g \in G\left[s'_j = CbO(s_j,g), s'_i = CbO(s_i,g)\right] \& \\ & \&\neg\exists m \in M\left[s'_j = CbO(s_j,m), s'_i = CbO(s_i,m)\right] \end{cases}.$$

Lemma 4 implies $\mathbb{P}\left[X_1 \leqslant Y_1 \mid X_0 \leqslant Y_0\right] = 1$.

Since $\langle A_i,B_i\rangle = \langle C_i,D_i\rangle$ for $\langle A_i,B_i\rangle \leq \langle A_j,B_j\rangle \leq \langle C_j,D_j\rangle \leq \langle C_i,D_i\rangle$ implies $\langle A_i,B_i\rangle = \langle A_j,B_j\rangle = \langle C_j,D_j\rangle = \langle C_i,D_i\rangle$, then by definitions it follows that

$$\mathbb{P}\left[X_t = Y_t \in E \mid X_0 = s_j \leqslant Y_0 = s_i\right] \geq \mathbb{P}\left[Y_t \in E \mid X_0 = s_j \leqslant Y_0 = s_i\right]. \tag{18}$$

Recall that for an integer-valued random variable Z, the equality $\mathbb{E}Z = \sum_{t=0}^{\infty} \mathbb{P}\left[Z > t\right]$ is fulfilled. Now $X_t \notin E \Leftrightarrow T_i(E) > t$ and $Y_t \notin E \Leftrightarrow T_j(E) > t$.

Therefore, equation (18) implies

$$\mathbb{P}\left[T_j(E) > t \mid X_0 = s_j, Y_0 = s_i\right] \leq \mathbb{P}\left[T_i(E) > t \mid X_0 = s_j, Y_0 = s_i\right],$$

and the summation over t leads to the required result. □

4. Main result

In the following we'll assume $G' = \emptyset$. This is easily achieved by eliminating all the attributes common to all training objects.

Let's dichotomize the context, i.e., enrich the set of attributes by introducing an attribute for the negation $\neg m_j$ of every binary attributes $m_j \in M$. This construction often has a useful meaning: we want the absence of a attribute to be a new attribute, i.e., we propose dichotomic scaling of the context (according to [5]).

The enriched set of attributes will be denoted by M^+, and we denote its power by $2n = |M^+|$. Usually $2n \ll k = |G|$, which we will assume in the future. Enrich the training context to $I \subseteq G \times M^+$ by the rule:

$$gI\neg m_j \Leftrightarrow \neg(gIm_j).$$

Divide all transient states into 2 groups:

$$V = \{s = (\langle A, B\rangle < \langle C, D\rangle) : \exists m \in M^+ \, [m \in B]\} \tag{19}$$

and

$$W = \{s = (\langle A, B\rangle < \langle C, D\rangle) : \forall m \in M^+ \, [m \notin B]\}. \tag{20}$$

It is clear that the state $s_0 = (\bot < \top) \in W$. By lemma 5 for any $s_j \in W, \mathbb{E}T_j(E) \leq \mathbb{E}T_0(E)$.

By the definition of the set V and the lemma 5 for any $s_j \in V$ we have $\mathbb{E}T_j(E) \leq \mathbb{E}T_i(E)$, where $s_i = (\langle \{m\}', \{m\}''\rangle < \top) \in V$ for any $m \in B$ with $s_j = \langle A, B\rangle$.

Let's introduce an integer-valued random variable Z taking the value q on the event $\{X_q = (\bot = \bot), X_{q-1} \notin V, \ldots, X_1 \notin V, X_0 = s_0\}$, which determines the minimum number of steps of the algorithm 1 by states from $X_t \in W$ until we get $X_q = (\bot = \bot)$.

Lemma 6.

$$\mathbb{E}Z = \sum_{l=1}^{\infty} \mathbb{P}\left[Z \geq l\right] \leq (k + 2n) \cdot \left(\ln(2n) + \frac{1}{1 - e^{-1}}\right)$$

for context $I \subseteq G \times M^+$ *with* $2n = |M^+| \leq k = |G|$.

Proof. We divide the summands into disjoint subsets of $I_0 \sqcup \bigsqcup_{r=1}^{\infty} I_r$, where $I_0 = \{1 \leq l < (k + 2n) \cdot \ln(2n)\}$ and

$$I_r = \{(k + 2n) \cdot (\ln(2n) + r - 1) \leq l < (k + 2n) \cdot (\ln(2n) + r)\}.$$

It is clear that $\sum_{l=1}^{(k+2n)\cdot\ln(2n)-1} \mathbb{P}\left[Z \geq l\right] \leq (k + 2n) \cdot \ln(2n)$.

In order for the event $Z \geq l$ to occur, it is necessary that at least one attribute (out of $2n$) is selected, so that no example in the series of length l is selected in which this attribute is not present. Therefore, by Boole's inequality

$$\mathbb{P}\left[Z > l\right] \leq 2n \cdot \left(1 - \frac{1}{k + 2n}\right)^l.$$

For I_r

$$\sum_{l=(k+2n)\cdot(\ln(n)+r-1)}^{(k+2n)\cdot(\ln(2n)+r)-1} \mathbb{P}\left[Z > l\right] \leq \sum_{l=(k+2n)\cdot(\ln(2n)+r-1)}^{(k+2n)\cdot(\ln(2n)+r)-1} k \cdot \left(1 - \frac{1}{k+2n}\right)^{l} \leq$$
$$\leq (k+2n)\cdot 2n \cdot \left(1 - \frac{1}{k+2n}\right)^{(k+2n)\cdot(\ln(2n)+r-1)} \leq$$
$$\leq (k+2n)\cdot e^{\ln 2n} \cdot e^{-(\ln(2n)+r-1)} = (k+2n)\cdot e^{-r+1}.$$

The summation over r gives

$$\sum_{l=(k+2n)\cdot\ln(2n)}^{\infty} \mathbb{P}\left[Z \geq l\right] \leq (k+2n)\cdot \sum_{r=1}^{\infty} e^{-r+1} = \frac{k+2n}{1-e^{-1}}.$$

□

Let's denote the upper bound from lemma 6 by $tail$.

We consider disjoint events

$$H_l(s_j) = \{X_l = s_j \in V, X_{l-1} \notin V, \ldots, X_1 \notin V, X_0 = s_0\}. \tag{21}$$

We denote event $\{X_{t+l} \in E, X_{t+l-1} \notin E, \ldots, X_{l+1} \notin E\} \cap H_l(s_j)$ by $G_{t,l}(s_j)$, and the union $\bigsqcup_{s_j \in V} G_{t,l}(s_j)$ by $U_{t,l}$.

It is clear that we have a decomposition of the event into disjoint parts

$$\{X_{t+l} \in E, X_{t+l-1} \notin E, \ldots, X_0 = s_0\} =$$
$$= \bigsqcup_{s_j \in V} \left(\{X_{t+l} \in E, X_{t+l-1} \notin E, \ldots, X_{l+1} \notin E\} \cap H_l(s_j)\right) \sqcup$$
$$\sqcup \{X_{t+l} = (\bot = \bot), X_{t+l-1} \notin V, \ldots, X_1 \notin V, X_0 = s_0\}.$$

We need

$$\mathbb{E}T_0(E) = \sum_{m=1}^{\infty} m \cdot \mathbb{P}\left[X_m \in E, X_{m-1} \notin E, \ldots, X_1 \notin E \mid X_0 = s_0\right]. \tag{22}$$

It is clear that $\mathbb{E}T_0(E) = \mathbb{E}T_0'(E) + \mathbb{E}Z$, where $T_0'(E)$ is restriction of $T_0(E)$ on $\bigsqcup_{t=1}^{\infty} \bigsqcup_{l=1}^{\infty} G_{t,l}$.

Theorem 2. *For the dichotomized (enriched) training context $I \subseteq G \times M^+$ with $2n = |M^+| \leq k = |G|$ the upper bound on the average length of trajectories of algorithm 1 is*

$$\mathbb{E}T_0 \leq \frac{(k+2n)(k^2 + k(2n+1) + 4n^2 + 2n)}{2n(k^2+k+2n)} + \frac{(k+1)(k+2n)}{k^2+k+2n} tail.$$

Proof. Let's denote $R = \sum_{l=1}^{n} \frac{1}{k+2n}\left(T_{f_l} + T_{\neg f_l}\right)$, where $T_{f_l} = T_i(E)$ for $s_j = (\langle \{f_l\}', \{f_l\}'' \rangle < \top)$, and similarly for $T_{\neg f_l}$.

Then Markov property implies

$$\mathbb{E}T'_0(E) = \sum_{t=1}^{\infty}\sum_{l=1}^{\infty}(t+l)\cdot \mathbb{P}U_{t,l} =$$

$$= \sum_{t=1}^{\infty} t \cdot \sum_{s_j\in V} \mathbb{P}\left[X_t \in E, X_{t-1}\notin E, \ldots, X_1 \notin E \mid X_0 = s_j\right] \cdot \sum_{l=1}^{\infty}\mathbb{P}\left[H_l(s_j)\right] +$$

$$+ \sum_{l=1}^{\infty} l \cdot \sum_{s_j\in V}\mathbb{P}\left[H_l(s_j)\right]\cdot \sum_{t=1}^{\infty}\mathbb{P}\left[X_t \in E, X_{t-1}\notin E, \ldots, X_1 \notin E \mid X_0 = s_j\right] \le$$

$$\le \sum_{s_j\in V}\mathbb{E}T_j(E)\cdot \mathbb{P}\left[X_1 = s_j \mid X_0 = s_0\right] + \sum_{s_j\in V}\sum_{l=1}^{\infty} l \cdot \mathbb{P}\left[H_l(s_j)\right] \le$$

$$\le \mathbb{E}R + \frac{k+2n}{2n},$$

where the last term is the average of geometrically distributed random variable of the time before first selection of some attribute.

The Law of Total Probability and lemma 5 imply

$$\mathbb{E}T_{f_l} \le 1 + \sum_{i=1}^{n}\frac{1}{k+2n}\left(\mathbb{E}T_{f_i} + \mathbb{E}T_{\bar{f}_i}\right) - \frac{1}{k+2n}\cdot \mathbb{E}T_{\bar{f}_l} + \frac{k}{k+2n}\mathbb{E}T_0(E).$$

Therefore,

$$\mathbb{E}R \le \frac{2n}{k+2n}\left[1 + \mathbb{E}R + \frac{k}{k+2n}\mathbb{E}T_0(E)\right] - \frac{1}{k+2n}\mathbb{E}R.$$

Hence,

$$\frac{k+1}{k+2n}\mathbb{E}R \le \frac{2n}{k+2n} + \frac{2nk}{(k+2n)^2}\mathbb{E}T_0(E).$$

Substitute $\mathbb{E}R \le \frac{2n}{k+1} + \frac{2nk}{(k+1)(k+2n)}\mathbb{E}T_0(E)$ into

$$\mathbb{E}T_0(E) \le \mathbb{E}R + \frac{k+2n}{2n} + tail,$$

and obtain

$$\frac{k^2+k+2n}{(k+1)(k+2n)}\mathbb{E}T_0(E) \le \frac{2n}{k+1} + \frac{k+2n}{2n} + tail,$$

which leads to the required result. □

5. Conclusion

In this article we have presented a significant advancement in solving the open problem of the VKF method about finding a polynomial upper bound on the average length of trajectories of a coupling Markov chain - the average time of computation by the probabilistic algorithm 1, which generates concepts of the training context for knowledge extraction. Only special cases, such as Boolean algebra and linear order, were investigated earlier. The important step is based on the dichotomic scaling of a training context.

Combining the new result with the previously obtained polynomial lower bound on the sufficient number of concepts, we obtain a fully polynomial scheme for extracting knowledge using a binary similarity operation implemented in the VKF-method.

Experimental studies of author's PhD student L.A. Yakimova demonstrate that probabilistic approach to FCA-based knowledge extraction (in combination with "Counterexample Forbidding Condition") is practically not subject to the phenomenon of over-fitting (through generation of 'phantom' candidates), unlike the classical JSM-method.

Acknowledgments

The author thanks his colleagues from Dorodnicyn Computing Center of Federal Research Center "Computer Science and Control" of Russian Academy of Sciences for support and useful discussions. The author is grateful to his PhD student Lyudmila A. Yakimova for long-term cooperation that stimulated the described research.

References

[1] V. K. Finn, J.S.Mill's inductive methods in artificial intelligence systems I, Scientific and Technical Information Processing 38 (2011) 385–402. doi:10.3103/S0147688211060037.

[2] V. K. Finn, J.S.Mill's inductive methods in artificial intelligence systems II, Scientific and Technical Information Processing 39 (2012) 241–260. doi:10.3103/S0147688212050036.

[3] J. S. Mill, A System of Logic: Ratiocinative and Inductive, John W. Parker, London, 1843.

[4] S. O. Kuznetsov, Machine learning on the basis of formal concept analysis, Automation and Remote Control 62 (2001) 1543–1564. doi:10.1023/A:1012435612567.

[5] B. Ganter, R. Wille, Formal Concept Analysis: Mathematical Foundations, Springer, Berlin, Heidelberg, 1999. doi:10.1007/978-3-642-59830-2.

[6] S. O. Kuznetsov, S. A. Obiedkov, Algorithms for the construction of concept lattices and their diagram graphs, in: L. D. Raedt, A. Siebes (Eds.), Proceedings of the 5th Conference on Principles of Data Mining and Knowledge Discovery, volume 2168 of *Lecture Notes in Artificial Intelligence*, Springer, Berlin, Heidelberg, 2001, pp. 289–300. doi:10.1007/3-540-44794-6.

[7] D. V. Vinogradov, Existence of large sublattices isomorphic to boolean algebra in a candidate lattice, Autom. Doc. Math. Linguist. 57 (2023) 101–103. doi:10.3103/S0005105523020097.

[8] D. V. Vinogradov, Accidental formal concepts in the presence of counterexamples, in: S. O. Kuznetsov, B. W. Watson (Eds.), Proceedings of International Workshop on Formal Concept Analysis for Knowledge Discovery, volume 1921 of *CEUR Workshop Proceedings*, HSE, Moscow, Russia, 2017, pp. 104–112.

[9] L. A. Yakimova, Experimental investigation of behaviour of solvers based on binary similarity operation, Master's thesis, Russian State University for Humanities, Moscow, Russia, 2020. In Russian.

[10] D. V. Vinogradov, VKF-method of hypotheses generation, in: Proceedings of the 3rd International Conference on Analysis of Images, Social Networks and Texts (AIST'2014), volume 436 of *Communications in Computer and Information Science*, 2014, pp. 237–248. doi:10.1007/978-3-319-12580-0_25.

[11] S. O. Kuznetsov, A fast algorithm for computing all intersections of objects from an arbitrary semilattice, Nauch.-Tekh. Inf. Ser.2 27 (1993) 17–20.

[12] D. V. Vinogradov, Algebraic machine learning: Emphasis on efficiency, Automation and Remote Control 83 (2022) 831–846. doi:10.1134/S0005117922060029.

[13] J. G. Kemeny, J. L. Snell, Finite Markov Chains, Undergraduate Texts in Mathematics, 1 ed., Springer, New York, 1976. Originally published by Van Nostrand Publishing Company, 1960.

[14] D. V. Vinogradov, Markov chains, law of total probability, and recurrence relations, Autom. Doc. Math. Linguist. 57 (2023) 68–72. doi:10.3103/S0005105523010090.

A Note on Counting Basic Choice Functions with Formal Concept Analysis

Dmitry I. Ignatov

HSE University, Moscow, Russia

Abstract

The paper aims at not only counting how many basic choice functions exist on a finite set of alternatives (all, non-empty, single-element valued) but shows how to do this with the help of Formal Concept Analysis. Moreover, we introduce the contextual representation of a choice function by considering the formal context of its map from 2^A to 2^A. We also characterise these contexts as nominal scales of a certain size and build a lattice of all choice functions with their help. Last but not least, we study the asymptotic behaviour of those obtained and new counting formulas that do not have a closed form.

Keywords

Choice function, Concept Lattice, Combinatorics, Asymptotic analysis

1. Introduction

Choice Theory is formalised with the help of Order Theory [1, 2] and has applications not only in Social Sciences but also in Artificial Intelligence, e.g. to model and learn preferences of agents [3, 4]. In particular, it deals with set-valued functions defined on a set of alternatives, i.e. variants that an individual or (rational) agent can choose based on her preferences or utility function [5, 6, 1].

In this paper, inspired by earlier works on choice functions and Lattice Theory [5, 6, 2] (including Formal Concept Analysis (FCA) as its applied branch [7, 8]), we characterise concept lattices induced by point-wise representations of choice functions considered as formal contexts and count basic choice functions (all, non-empty, single-element valued) for a fixed number of alternatives.

The previous work of Monjardet and Raderanirina [2] studies the space of all choice functions fulfilling certain axioms (called heredity, concordance, and outcast), which forms lattices if one of the axioms is fulfilled. The works of Revenko and Kuznetsov [9, 8] consider various axioms on set functions as (formal) attributes and perform attribute exploration [10] (an interactive semi-automatic procedure of hypotheses generation in terms of attribute implications and checking them by an expert) with functions on sets up to four elements. Not only choice functions were considered in [9, 8] since the choice functions are intensive, but extensity property was also included.

FCA4AI 2023

dignatov@hse.ru (D. I. Ignatov)
https://www.hse.ru/en/staff/dima (D. I. Ignatov)
0000-0002-6584-8534 (D. I. Ignatov)

Other related works on FCA and Choice Theory include learning individual and collective preferences [11], enchaining consensus voting procedures [12], for example, in consensus clustering [13], studying games on concept lattices [14, 15], and attribute ranking in formal concepts with Shapley values [16].

The paper is organised as follows. In Section 2 we give basic definitions from FCA and for considered families of choice functions. Section 3 contains our main results split in three subsections on the proposed conceptual representation of choice functions, three counting formulas, and their asymptotic behaviour, respectively. The last section concludes the paper.

2. Basic Notions

2.1. Formal Concept Analysis

We recall several definitions from Formal Concept Analysis [7], an applied branch of modern Lattice Theory. We reproduce basic definitions from our tutorial [17], for more details see also textbook [18].

A *formal context* $\mathbb{K} = (G, M, I)$ consists of two sets G and M and a relation I between G and M. The elements of G are called the *objects* and the elements of M are called the *attributes* of the context. The notation gIm or $(g, m) \in I$ means that the object g has attribute m.

A special type of context defined on any set S is used in the next section: the *nominal scale* $\mathbb{N}_S := (S, S, =)$.

For $A \subseteq G$ and $B \subseteq M$, let

$$A' := \{m \in M \mid (g, m) \in I \text{ for all } g \in A\}$$

$$B' := \{g \in G \mid (g, m) \in I \text{ for all } m \in B\}.$$

These operators are called *derivation operators* or *concept-forming operators* for $\mathbb{K} = (G, M, I)$.

Proposition 1. *Let (G, M, I) be a formal context, for subsets $A, A_1, A_2 \subseteq G$ and $B \subseteq M$ we have*

1. $A_1 \subseteq A_2 \Rightarrow A_2' \subseteq A_1'$ *(antimonotony of* $'$*),*
2. $A_1 \subseteq A_2 \Rightarrow A_1'' \subseteq A_2''$ *(monotony of* $''$*),*
3. $A \subseteq A''$ *(extensity of* $''$*),*
4. $A' = A'''$ *(hence,* $A'''' = A''$*, i.e. idempotency of* $''$*),*
5. $(A_1 \cup A_2)' = A_1' \cap A_2'$,

Similar properties hold for subsets of attributes.

Note that traditionally $\{g\}'$ and $\{m\}'$ are written as g' and m' for brevity.

For $\mathbb{K} = (G, M, I)$, the operators $(\cdot)'' : 2^G \to 2^G$, $(\cdot)'' : 2^M \to 2^M$ are closure operators, i.e. idempotent, extensive, and monotone.

A *formal concept* of a formal context $\mathbb{K} = (G, M, I)$ is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. The sets A and B are called the extent and the intent of the formal concept (A, B), respectively. The *subconcept-superconcept relation* is given by $(A_1, B_1) \leq (A_2, B_2)$ iff $A_1 \subseteq A_2$ $(B_2 \subseteq B_1)$.

The set of all formal concepts of a context $\mathbb{K}$ together with the order relation $\leq$ forms a complete lattice called the *concept lattice* of $\mathbb{K}$ and denoted by $\underline{\mathfrak{B}}(\mathbb{K})$.

2.2. Choice Functions

A choice function on a set A is defined as map $C : 2^A \to 2^A$ such that $C(A) \subseteq A$ (intensity property).

In what follows, we adopt terminology from [1]. Let $\mathscr{A}$ be the set of all non-empty subsets of A, while $\mathscr{C}$ be the set of all choice functions on A. The subset $\mathscr{C}^+$ of $\mathscr{C}$ contains only non-empty choice functions, i.e. $C(X) \neq \emptyset$ for all $X \in \mathscr{A}$.

The set of all single-valued functions $\widehat{\mathscr{C}}$ contains $\hat{C}$ such that $|\hat{C}(X)| = 1$ for all $X \in \mathscr{A}$.

3. Main Results

3.1. Conceptual Representation

Let us form the context representing a choice function as follows $\mathbb{K}_C := (G, M, I)$ with $G := 2^A$, $M := 2^A$, $I :=\subseteq$, where for $g \in G$, $m \in M$, gIm iff $C(g) = m$. It is clear that the domain of C $dom(C) = G$, while $range(C) \subseteq M$.

Contexts representing non-empty and single-valued functions are denoted $\mathbb{K}_{C^+} := (\mathscr{A}, \mathscr{A}, \subseteq)$ and $\mathbb{K}_{\hat{C}} := (\mathscr{A}, \mathscr{A}, \subseteq)$, respectively, where for the last context $gIm \iff \hat{C}(g) = m$ and $|m| = 1$.

Proposition 2. *Let $C \in \mathscr{C}$, $C^+ \in \mathscr{C}^+$, $\hat{C} \in \widehat{\mathscr{C}}$ and $|A| = n$, then the concept lattices of $\mathbb{K}_C = (2^A, 2^A, \subseteq)$, $\mathbb{K}_{C^+} = (\mathscr{A}, \mathscr{A}, \subseteq)$ and $\mathbb{K}_{\hat{C}} = (\mathscr{A}, \mathscr{A}, \subseteq)$ are isomorphic to the lattices of nominal scales $N_n = ([k], [k], =)$[1] where $k = |range(F)|$ for $F \in \{C, C^+, \hat{C}\}$ and 1) $1 \leq k \leq 2^n$, 2) $n \leq k \leq 2^n - 1$, and 3) $k = n$, respectively.*

Proof. 1) $range(C)$ may vary from $\{\emptyset\}$ set to 2^A, which means that the number of $m \in 2^A$ such that $m' \neq \emptyset$ varies from 1 to 2^n. Equality 2) follows from the condition $\forall g \in \mathscr{A} : |g| = 1 \Rightarrow g' = \{g\}$ (by intensity of $C(g)$). Equality 3) follows from the previous condition and condition $\forall g \in \mathscr{A}\, \exists a \in A : g' = \{a\} \wedge a \in g$.

$\square$

The interpretation of concepts in such lattices is straightforward. Let $m \in G = 2^A$, then (m', m) contains the image m as the intent and its preimage m' (or the fibre $C^{-1}(\{m\})$, the set all of sets that mapped to $\{m\}$) as the extent. Note that $\{m\}'' = \{m\}$ and there are no other concepts than (m', m), (G, G') and (M', M).

The following example is inspired by our previous work on how university entrants are choosing their departments [19].

Example 1. *Let us consider a set A with three alternatives a_1 (Computer Science faculty), a_2 (Mathematical faculty), and a_3 (Faculty of Economics). It is known that if an individual S has preferences represented by a binary relation P, then they can be rationalised by a choice function under certain conditions [1]. Since the choice is not necessarily effective (a single-alternative outcome), our individual may choose two alternatives $C(A) = \{a_1, a_2\}$ out of three.*

Definitely, she loves Mathematics and Computer Science so the choice between those two is not final, $C(\{a_1, a_2\}) = \{a_1, a_2\}$. When only a single faculty out of the last two is available, she chooses

[1]We use $[n]$ for $\{1, 2, \dots, n\}$

$\mathbb{K}_C$	$\emptyset$	a_1	a_2	a_3	a_1, a_2	a_1, a_3	a_2, a_3	a_1, a_2, a_3
$\emptyset$	×							
a_1		×						
a_2			×					
a_3	×							
a_1, a_2					×			
a_1, a_3		×						
a_2, a_3							×	
a_1, a_2, a_3					×			

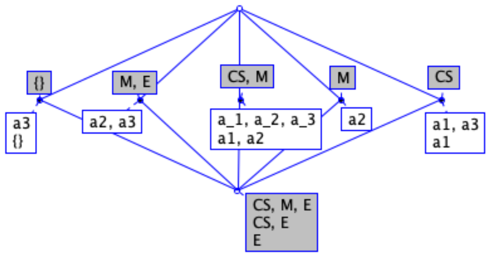


Figure 1: An example context for $\mathbb{K}^C$ and its concept lattice diagram.

it. However, when only the faculty of Economics is offered, she refuses and probably takes a year gap (it might be a very pity that there is no choice among the favourite faculties). However, when educational tracks for mathematics and economics are compared, she might decide to apply both. So, the choices might seem to be not fully rational (in terms of common sense), but they are in line with the definition of $C(\cdot)$.

The line diagram of the corresponding concept lattice $\underline{\mathfrak{B}}(\mathbb{K}_C)$ on the left in Fig. 1 is drawn in Concept Explorer. We use the so-called reduced labelling when nodes (representing concepts) are labelled with object names when objects are first time added to the extent of a concept (when we go from the bottom concept to the topmost one) and attribute names when attributes are first time added to the intent of a concept (when we go in top-to-bottom direction). Note that we use shorthand CS, M, and E in the attribute labels (the latter denote choices on all alternative subsets), and a_1, a_2, and a_3 in the object labels (the latter denote the subsets of all the alternatives).

Note that our attributes are sets of alternatives and $\{a_3\}, \{a_1, a_3\}$, and $\{a_1, a_2, a_3\}$ can be eliminated from the $\mathbb{K}_C$ without affecting the lattice structure. The obtained concept lattice is isomorphic to the so-called diamond lattice M_5.

The lattice of a choice function can be defined via point-wise intersection and union. Let us order objects of $G = 2^A$ first by their cardinality and lexicographically for sets of equal cardinality such that $g_0 = \emptyset, \ldots, g_{2^n-1} = A$. Now, every choice function C is represented by its point-wise vector of images $im(C) = (\bigcup g_0', \bigcup g_1', \ldots, \bigcup g_{2^n-1}')$[2]. Note that $g_0' = \{\emptyset\}$.

For the example in Fig. 1, we have $im(C) = (\emptyset, \{a_1\}, \{a_2\}, \emptyset, \{a_1, a_2\}, \{a_1\}, \{a_2, a_3\}, \{a_1, a_2\})$.

For two functions C_1 and C_2, the supremum and infimum of their point-wise vectors of images

$$im(C_1) = (\bigcup g_0^{I_1}, \bigcup g_1^{I_1}, \ldots, \bigcup g_{2^n-1}^{I_1}) \text{ and } im(C_2) = (\bigcup g_0^{I_2}, \bigcup g_1^{I_2}, \ldots, \bigcup g_{2^n-1}^{I_2})$$

(primes are taken in the respective contexts) are defined as follows:

$$im(C_1) \bigvee im(C_2) = (\bigcup g_i^{I_1} \cup \bigcup g_i^{I_2})_{i=0}^{2^n-1},$$

$$im(C_1) \bigwedge im(C_2) = (\bigcup g_i^{I_1} \cap \bigcup g_i^{I_2})_{i=0}^{2^n-1}.$$

[2]we use $\bigcup$ as a set unfolding operation since $g_i' = \{m_j\}$ and $C(g_i) \equiv \bigcup g_i'$

Their existence is guaranteed by set intersection and union on images of choice functions.

Let $A = [n]$, then triple $\mathfrak{L}(C) = (Im(\mathscr{C}), \bigvee, \bigwedge)$[3] forms a lattice with $\mathbf{0} = (\emptyset)_{i=0}^{2^n-1}$ and $\mathbf{1} = (\emptyset, \{1\}, \dots, [n])$, while $\mathfrak{S} = (Im(\mathscr{C}^+), \bigwedge)$ is an upper-semilattice and $\mathfrak{A} = (Im(\widehat{\mathscr{C}}), \leq)$ forms an antichain with respect to the point-wise set inclusion of components $\leq$ $(\forall \widehat{C_1}, \widehat{C_2} \in \widehat{\mathscr{C}} : im(\widehat{C_1}) \leq im(\widehat{C_2}) \iff \bigcup g_i^{I_1} \subseteq \bigcup g_i^{I_2}$ for $i \in [2^n - 1])$.

3.2. Counting Cardinalities

Let us prove the following proposition on the cardinality of $\mathscr{C}_n$, $\mathscr{C}_n^+$, $\widehat{\mathscr{C}_n}$ where $|A| = n$.

Proposition 3.

$$|\mathscr{C}_n| = 2^{n2^{n-1}} \tag{1}$$

$$|\mathscr{C}_n^+| = \prod_{k=1}^{n} (2^k - 1)^{\binom{n}{k}} \tag{2}$$

$$|\widehat{\mathscr{C}_n}| = \prod_{k=1}^{n} k^{\binom{n}{k}} \tag{3}$$

Note that (1) and (2) have been proven in [20] according to [1] (where they are given without proof). We give our proof of (1) and (2) with the help of FCA.

Proof. 1) Let us consider $\mathbf{1} = (\emptyset, \{1\}, \dots, [n])$ it corresponds to $\mathbb{K}_{C_{id}} = (2^A, 2^A, I_{id})$, where $C_{id}(X) = X$ for $X \subseteq A$ and $I_{id} :==$. For each other choice function, $im(C)$ is below $\mathbf{1}$ in the lattice $\mathfrak{L}(\mathscr{C})$, which means that $\bigcup g_i{}' \subseteq \bigcup g_i^{I_{id}}$, where $'$ is taken in the $\mathbb{K}_C$. Thus each row of $\mathbb{K}_C$ has $|2^{\bigcup g_i^{I_{id}}}|$ variants and the choice of each row is independent (we are ready for the product rule).

$$\prod_{k=0}^{2^n-1} 2^{|\bigcup g_i^{I_{id}}|} = \prod_{X \subseteq A} 2^{|X|} = \prod_{k=0}^{n} 2^{k\binom{n}{k}}$$

The last step is due to the presence of each set of size k $\binom{n}{k}$ times. The sum $\sum_{k=0}^{n} k\binom{n}{k}$ equals $n2^{n-1}$.

2) Now, we are not allowed to consider $\bigcup g_i^{I_{id}} = \emptyset$, which implies subtraction of 1 (i.e. $2^k - 1$) when counting variants for the choice of a row in the context $\mathbb{K}_{C^+} = (\mathscr{A}, \mathscr{A}, \subseteq)$. Here $I_{id} \subseteq \mathscr{A} \times \mathscr{A}$ so g_0 (also m_0) is excluded and the product starts with $k = 1$.

3) Here, compared to the previous case, since we can choose only single-element sets among all $m \in A$, $2^k - 1$ is simply replaced by k.

□

Note that Monjardet and Raderanirina [2] claim that the lattice of all choice functions on a set of alternatives A is Boolean (i.e. atomistic and distributive) with $n2^{n-1}$ atoms, which directly implies the proof of (1).

[3] $Im(\mathscr{C}) = \{im(C) \mid C \in \mathscr{C}\}$

3.3. Asymptotic Analysis

The values represented by equations 1 and 2 have no closed-form formulas but are smaller than the size of the whole space of choice functions. Our goal here is to figure out their asymptotic behaviour to better understand how the sizes of the posets, $|\mathscr{C}_n^+|$, $|\mathscr{C}_n|$, and $|\widehat{\mathscr{C}_n}|$, interrelated.

Proposition 4.

$$\log_2 |\mathscr{C}_n^+| = n2^{n-1} + O(2^n n^{-1/2})$$

Proof. Let us apply $\log_2$ to the product (2).

$$\log_2 |\mathscr{C}_n^+| = \sum_{k=1}^{n} \binom{n}{k} \log_2 2^k + \sum_{k=1}^{n} \binom{n}{k} \log_2(1 - 1/2^k)$$

The first sum equals (1), while the second is more laborious since it has no closed form. Since $\log_2 x \le x - 1$ for all $x > 0$, we obtain

$$\sum_{k=1}^{n} \binom{n}{k} \log_2(1 - 1/2^k) \le \sum_{k=1}^{n} \binom{n}{k} (-1/2^k) = -\left(\frac{3}{2}\right)^n + 1$$

Since $1/2 \le (1 - 1/2^k) < 1$ for $k \ge 1$, we have $-2^n \le \sum\limits_{k=1}^{n} \binom{n}{k} \log_2(1 - 1/2^k)$. However, we can do better with the lower bound if pull out the maximal binomial coefficient, i.e. the middle (or central) binomial coefficient.

$$\sum_{k=1}^{n} \binom{n}{k} \log_2(1 - 1/2^k) \ge \binom{n}{\lfloor n/2 \rfloor} \sum_{k=1}^{n} \log_2(1 - 1/2^k) \ge \binom{n}{\lfloor n/2 \rfloor} \sum_{k=1}^{\infty} \log_2(1 - 1/2^k)$$

$$\sum_{k=1}^{\infty} \log_2(1 - 1/2^k) = \log_2 \prod_{k=1}^{\infty} (1 - 1/2^k) = \log_2 \phi(1/2) \approx -1.79192, \text{ where}$$

$\phi(q) \equiv (q)_\infty \equiv (q;q)_\infty = \prod\limits_{k=1}^{\infty} (1 - q^k)$ is the Euler function [21], and $(q)_\infty$ and $(q;q)_\infty$ are q-Pochhammer symbols [22].

The variable term $\binom{n}{\lfloor n/2 \rfloor}$ is $O(2^n n^{-1/2})$ since for even n, we have $\binom{n}{n/2} = \sqrt{\frac{2}{\pi n}} 2^n \big(1 + O(1/n)\big)$ [23] and the following inequalities are known $\frac{2^n}{n} \le \binom{n}{\lfloor n/2 \rfloor} < \sqrt{2/\pi} \cdot 2^n n^{-1/2}$ [24] for integer $k \ge 0$.

□

Proposition 5.

$$\lim_{n \to \infty} \frac{|\mathscr{C}_n^+|}{|\mathscr{C}_n|} = \prod_{k=1}^{\infty} (1 - \frac{1}{2^k})^{\binom{n}{k}} \textit{ diverges to zero.}$$

Proof. From the proof of the previous proposition it follows that

$$\phi(1/2)^{\sqrt{2/\pi} \cdot 2^n n^{-1/2}} \le \prod_{k=1}^{n} (1 - \frac{1}{2^k})^{\binom{n}{k}} \le 2^{-\left(\frac{3}{2}\right)^n + 1} \text{ where } \phi(1/2) \approx 0.2888.$$

When n tends to ∞, both sides tend to 0, and since no terms of the partial product are zeros, the whole product is said to diverge to zero [25, 26].

$\square$

Proposition 6.

$$\log_2 |\widehat{\mathscr{C}_n}| = \Theta(2^n \log_2 n)\ .$$

Proof. To prove the statement we need to show that there are constants $c_1, c_2 > 0$, such that $c_1 2^n \log_2 n \leq \log_2 |\widehat{\mathscr{C}_n}| \leq c_2 2^n \log_2 n$ for all $n > n_0$.

We can pull out the largest value that $\log_2 n$ takes

$$\log_2 |\widehat{\mathscr{C}_n}| = \sum_{k=1}^{n} \binom{n}{k} \log_2 k \leq \log_2 n \sum_{k=1}^{n} \binom{n}{k} = (2^n - 1) \log_2 n.$$

For the lower bound we can split the sum into two parts as follows:

$$\sum_{k=1}^{n} \binom{n}{k} \log_2 k \geq \sum_{k=1}^{\lfloor n/2 \rfloor - 1} \binom{n}{k} \log_2 k + \sum_{k=\lfloor n/2 \rfloor}^{n} \binom{n}{k} \log_2 \frac{n}{2} \geq \sum_{k=\lfloor n/2 \rfloor}^{n} \binom{n}{k} \log_2 \frac{n}{2} \geq$$

$$\geq \log_2 \frac{n}{2} \sum_{k=1}^{n} \binom{n}{k} / 2 = \frac{1}{2} (\log_2 n - 1)(2^n - 1)\ .$$

$\square$

The last result can be sharpened to $\log_2 |\widehat{\mathscr{C}_n}| = 2^n \log_2 n\big(1 + O(1/log_2 n)\big)$. By changing k to $n-k$ we get $\sum\limits_{k=1}^{n} \binom{n}{k} \log_2 k = \sum\limits_{k=0}^{n-1} \binom{n}{k} \log_2 (n-k)$ and pull out $\log_2 n$, which gives us the term $(2^n - 1)\log_2 n$ and the remaining term is

$$\sum_{k=0}^{n-1} \binom{n}{k} \log_2 (1 - k/n) \leq -\frac{1}{n} \sum_{k=0}^{n-1} \binom{n}{k} k = -\sum_{k=0}^{n-1} \binom{n-1}{k-1} = 1 - 2^{n-1}\ .$$

4. Conclusion

Monjardet and Raderanirina [2] inform that not all spaces of choice functions with given properties have been explored in the sense that concrete counting formulae exist while a few beginning values are known.

For example, the lattice of choice functions satisfying hereditary axiom has size $|\mathfrak{L}(\mathscr{C}_n^H)| = (D_{n-1})^n$, where D_n is the n-th Dedekind number [2].

We get the new value $|\mathfrak{L}(\mathscr{C}_{10}^H)|$ with recently obtained D_9 [27][4] (with FCA):

$$286386577668298411128469151667598498812366^{10}.$$

We hope to continue this work on combinatorial properties of choice functions with FCA tools for their representation and counting and perform asymptotic analysis (if necessary).

[4]https://oeis.org/A000372

Acknowledgments

This study was implemented in the Basic Research Program's framework at HSE University.

We would like to thank the anonymous reviewers for relevant suggestions and the OEIS maintainers. We also would like to thank Lev P. Shibasov and Fuad T. Aleskerov for the inspirational lectures on Analysis and Choice Theory, respectively.

References

[1] F. Aleskerov, D. Bouyssou, B. Monjardet, Utility maximization, choice and preference, volume 16, Springer Science & Business Media, 2007.

[2] B. Monjardet, V. Raderanirina, Lattices of choice functions and consensus problems, Social Choice and Welfare 23 (2004) 349–382. doi:10.1007/s00355-003-0251-9.

[3] C. Boutilier, I. Caragiannis, S. Haber, T. Lu, A. D. Procaccia, O. Sheffet, Optimal Social Choice Functions, Artif. Intell. 227 (2015) 190–213. doi:10.1016/j.artint.2015.06.003.

[4] G. Pigozzi, A. Tsoukiàs, P. Viappiani, Preferences in artificial intelligence, Annals of Mathematics and Artificial Intelligence 77 (2016) 361–401. doi:10.1007/s10472-015-9475-5.

[5] H. Moulin, Choice functions over a finite set: A summary, Social Choice and Welfare 2 (1985) 147–160. doi:10.1007/BF00437315.

[6] M. Aizerman, F. Aleskerov, Voting operators in the space of choice functions, Mathematical Social Sciences 11 (1986) 201–242.

[7] B. Ganter, R. Wille, Formal Concept Analysis - Mathematical Foundations, Springer, 1999. doi:10.1007/978-3-642-59830-2.

[8] A. Revenko, S. O. Kuznetsov, Attribute exploration of properties of functions on sets, Fundam. Informaticae 115 (2012) 377–394. doi:10.3233/FI-2012-660.

[9] A. Revenko, S. O. Kuznetsov, Attribute exploration of properties of functions on ordered sets, in: M. Kryszkiewicz, S. A. Obiedkov (Eds.), Proceedings of the 7th International Conference on Concept Lattices and Their Applications, Sevilla, Spain, October 19-21, 2010, volume 672 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2010, pp. 313–324. URL: https://ceur-ws.org/Vol-672/paper28.pdf.

[10] B. Ganter, Attribute exploration with background knowledge, Theoretical Computer Science 217 (1999) 215–233. doi:10.1016/S0304-3975(98)00271-0, oRDAL'96.

[11] S. Obiedkov, Parameterized ceteris paribus preferences over atomic conjunctions under conservative semantics, Theoretical Computer Science 658 (2017) 375–390. doi:10.1016/j.tcs.2016.01.035.

[12] F. Domenach, A. Tayari, Implications of Axiomatic Consensus Properties, in: B. Lausen, D. Van den Poel, A. Ultsch (Eds.), Algorithms from and for Nature and Life, Springer International Publishing, Cham, 2013, pp. 59–67.

[13] A. Bocharov, D. Gnatyshak, D. I. Ignatov, B. G. Mirkin, A. Shestakov, A lattice-based consensus clustering algorithm, in: M. Huchard, S. O. Kuznetsov (Eds.), Proc. of CLA 2016, volume 1624 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016, pp. 45–56. URL: https://ceur-ws.org/Vol-1624/paper4.pdf.

[14] U. Faigle, M. Grabisch, A. Jiménez-Losada, M. Ordóñez, Games on concept lattices: Shapley

value and core, Discrete Applied Mathematics 198 (2016) 29–47. doi:https://doi.org/10.1016/j.dam.2015.08.004.

[15] K. Maafa, L. Nourine, M. S. Radjef, Algorithms for computing the Shapley value of cooperative games on lattices, Discret. Appl. Math. 249 (2018) 91–105. doi:10.1016/j.dam.2018.03.022.

[16] D. I. Ignatov, L. Kwuida, On Shapley value interpretability in concept-based learning with formal concept analysis, Ann. Math. Artif. Intell. 90 (2022) 1197–1222. doi:10.1007/s10472-022-09817-y.

[17] D. I. Ignatov, Introduction to formal concept analysis and its applications in information retrieval and related fields, in: P. Braslavski, N. Karpov, M. Worring, Y. Volkovich, D. I. Ignatov (Eds.), RuSSIR 2014, volume 505 of *Communications in Computer and Information Science*, Springer, 2014, pp. 42–141. doi:10.1007/978-3-319-25485-2_3.

[18] B. Ganter, S. A. Obiedkov, Conceptual Exploration, Springer, 2016. doi:10.1007/978-3-662-49291-8.

[19] N. Romashkin, D. I. Ignatov, E. Kolotova, How university entrants are choosing their department? mining of university admission process with FCA taxonomies, in: M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, J. C. Stamper (Eds.), Proceedings of the 4th International Conference on Educational Data Mining, Eindhoven, The Netherlands, July 6-8, 2011, www.educationaldatamining.org, 2011, pp. 229–234. URL: http://educationaldatamining.org/EDM2011/wp-content/uploads/proc/edm2011_paper31_short_Romashkin.pdf.

[20] V. Raderanirina, Treillis et agrégation de familles de Moore et de fonctions de choix, These de doctorat Université Paris 1 (2001).

[21] E. W. Weisstein, Euler Function, From MathWorld–A Wolfram Web Resource, 2023. URL: https://mathworld.wolfram.com/EulerFunction.html.

[22] B. C. Berndt, q-Series and Theta-Functions, Springer New York, New York, NY, 1991, pp. 11–86. doi:10.1007/978-1-4612-0965-2_2.

[23] D. E. Knuth, I. Vardi, R. Richberg, 6581. The Asymptotic Expansion of the Middle Binomial Coefficient, The American Mathematical Monthly 97 (1990) 626–630. URL: http://www.jstor.org/stable/2324649.

[24] Z.-H. Sun, Inequalities for binomial coefficients, 2013. arXiv:1310.0353.

[25] B. P. Demidovich, Problems in Mathematical Analysis, American First Edition ed., Mir Publishers, 1989.

[26] H. Jeffreys, B. Jeffreys, Methods of Mathematical Physics, Cambridge Mathematical Library, 3 ed., Cambridge University Press, 1999. doi:10.1017/CBO9781139168489.

[27] C. Jäkel, A computation of the ninth Dedekind Number, 2023. arXiv:2304.00895.

Dependency Covers from an FCA Perspective

Jaume Baixeries[1,*,†], Victor Codocedo[2,**,‡], Mehdi Kaytoue[3,***,§] and Amedeo Napoli[4,¶]

[1] *Universitat Politècnica de Catalunya, Barcelona, Catalonia*
[2] *Instituto para la Resiliencia ante Desastres, Chile*
[3] *Université de Lyon. CNRS, INSA-Lyon, LIRIS, Lyon, France*
[4] *Université de Lorraine, CNRS, LORIA, Nancy, France*

Abstract

Implications in Formal Concept Analysis (FCA), Horn clauses in Logic, and Functional Dependencies (FDs) in the Relational Database Model, are very important dependency types in their respective fields. Moreover, they have been proved to be equivalent from a syntactical point of view. Then notions and algorithms related to one dependency type in a field can be reused and applied to another dependency type in the other field. One of these notions is that of *cover*, also known as a *base* or *basis*, i.e., a compact representation of a complete set of implications, FDs, or Horn clauses. Although the notion of cover exists in the three fields, the characterization and the related uses of a cover are different. In this paper, we study and compare, from an FCA perspective, the principles on which rely the most important covers in each field. Finally, we discuss some open questions that are of interest in the three fields, and especially to the FCA community.

Keywords

Functional dependencies, Implications, Horn Clauses, Dependency Covers, Closure

1. Introduction and Motivation

A **dependency** is a relation between sets of attributes in a dataset. In this paper, they are represented as $X \rightarrow Y$, where the type of the subsets of attributes X and Y, and the semantics of $\rightarrow$ may vary w.r.t. the context. There are many different kinds of dependencies: complete and comprehensive surveys, from a Relational Database Theory perspective, can be found in [1] and in [2]. Here, we focus on those dependencies that follow the so called Armstrong axioms, this is, reflexivity, augmentation and transitivity, which appear in different fields of computer science: *functional dependencies* (FDs) in the Relational Database Model, *Horn clauses* in logic and logic programming, and *implications* in Formal Concept Analysis.

Functional dependencies [3] are of paramount importance in the Relational Database Model (RDBM), where they are used to express constraints or rules that need to hold in a database, to help the design of a database or to check for errors and inconsistencies. A set of **Horn clauses** [4] is a special case of Boolean functions that are crucial in logic programming

FCA4AI (Eleventh Edition) "What can FCA do for Artificial Intelligence?" co-located with IJCAI 2023, Macao, China, August 20 2023

jbaixer@cs.upc.edu (J. Baixeries); victor.codocedo@gmail.com (V. Codocedo); mehdi.kaytoue@gmail.com (M. Kaytoue); amedeo.napoli@loria.fr (A. Napoli)
0000-0003-2896-2247 (J. Baixeries); 0000-0002-6360-2039 (V. Codocedo); 0000-0002-1569-5242 (M. Kaytoue); 0000-0001-5236-9561 (A. Napoli)

[5, 6] and artificial intelligence (see [7] for a detailed explanation). **Implications** are at the core of Formal Concept Analysis (FCA) where they are used to model and deduce relevant information that is contained in a formal context [8, 9].

Although they appear in different fields, these three constructions have been applied on different kinds of data and have been used for different purposes. They also all share the same axioms, which means that, from a syntactical point of view, they are all equivalent. More specifically, the equivalence between functional dependencies and Horn clauses is presented in [10, 11] (see also [7] for a more detailed explanation). The equivalence between functional dependencies and implications is explained in [8] and the equivalence between implications and Horn clauses is explained in Section 5.1 in [9] as well as in [12]. These equivalences allow us to talk in a generic way of **Armstrong dependencies** or, simply, **dependencies**.

One of the consequences of this equivalence is the transversality of concepts, problems and algorithms between these three fields. One of the most typical examples is the decision problem of the **logical implication** which consists in, given a set of dependencies Σ and a single dependency σ, to determine whether σ is logically entailed by Σ, that is, $\Sigma \models \sigma$. Entailment means that σ can be derived from Σ by the iterative application of the Armstrong axioms. This problem is named **implication problem** in the RDBM [13, 14] and FCA fields, and **deduction** in logic [7]. It is of capital interest in all three fields. In the RDBM it allows to test whether two different sets of functional dependencies are equivalent [2], and it also allows to compute a more succinct set of functional dependencies, which is relevant to assist the design process of databases [15, 16]. In logic the deduction problem is used to check whether a logical expression is consistent w.r.t. a knowledge base and to compute the prime implicants of a Horn function [7]. In Formal Concept Analysis this problem is used, for instance, in attribute exploration [9], which consists in creating a data table (context) in agreement w.r.t. a set of attributes and a set of objects, and also for computing the Duquenne-Guigues basis [17].

Roughly speaking, the computation of the logical implication problem $\Sigma \models X \rightarrow Y$ is performed by iterating over Σ and applying the Armstrong axioms to infer new dependencies until a positive or negative answer is found. However, this problem can be reduced to the computation of the **closure** of X with respect to Σ ($\mathrm{closure}_\Sigma(X)$). This closure returns the largest set such that $\Sigma \models X \rightarrow \mathrm{closure}_\Sigma(X)$ holds. Therefore, the implication problem $\Sigma \models X \rightarrow Y$ boils down to testing whether $Y \subseteq \mathrm{closure}_\Sigma(X)$.

As an example of transversality, the algorithm that computes $\mathrm{closure}_\Sigma(X)$ appears in most of the main database textbooks, where it is called **closure** [14, 13, 3], and also in logic, where it is called **forward chaining** [7], while in FCA the same algorithm that first appeared in the RDBM is discussed and reused in [9].

Another "transversal notion" which is present in all three fields is the notion of **cover**. In general terms, it is not suitable to handle the set Σ of all dependencies that hold, because of its potential large size, but rather a subset of Σ that contains the same information and that is significantly smaller in size. By *"containing the same information"* we mean that this subset may generate, thanks to the application of the Armstrong axioms, the complete set Σ. This compact and representative subset is called "cover" in the RDBM, "basis" in FCA and "set of prime implicants" in logic. Moreover, each field has defined and used a different kind of cover. While in the RDBM this base is the Canonical-Direct Basis or the Minimal Cover, the cover of choice in FCA is the so-called Duquenne-Guigues basis.

Both the implication problem and the problem of computing a cover are related: the implication problem is used in the algorithm **Canonical Basis** (Algorithm 16, page 103 in [9]) to compute the Duquenne-Guigues basis, and it is also used in the algorithm **Direct** (Section 5.4, Chapter 5 in [13]) which is used to compute the Minimal Cover. Again, the transversality of the Armstrong dependencies appears in a general concept (computing a cover) but in different forms (Duquenne-Guigues basis and Minimal Cover).

This paper is a short version of the paper *Three Views on Dependency Covers from an FCA Perspective* that was accepted at the ICFCA 2023 (International Conference on Formal Concept Analysis). The purpose of this study is to present a discussion of the main different covers used in the RDBM, in Logic, and in FCA from the perspective of the formal concept analysis community. To do so, we review three different main covers that appeared in the literature.

The paper is organized as follows. Section 2 provides the necessary definitions needed in the paper. Section 3 includes a detailed comparison of the main covers. Finally, Section 4 concludes the paper and proposes an extensive discussion about these different and important covers.

2. Definitions

In this section we introduce the definitions used in this paper. We do not provide the references for all of them because they can be found in all the textbooks and papers related to the RDBM, Logic and FCA.

As explained in the introduction, implications[8], functional dependencies [13] and Horn clauses [4] are dependencies between sets of attributes, which are equivalent from a syntactical point of view, since they are in agreement with the Armstrong axioms.

Definition 2.1. *Given set of attributes $\mathcal{U}$, for any $X, Y, Z \subseteq \mathcal{U}$, the Armstrong axioms are:*

1. ***Reflexivity**: If $Y \subseteq X$, then $X \rightarrow Y$ holds.*
2. ***Augmentation**. If $X \rightarrow Y$ holds, then $XZ \rightarrow YZ$ holds.*
3. ***Transitivity**. If $X \rightarrow Y$ and $Y \rightarrow Z$ hold, then $X \rightarrow Z$ holds.*

When we write that a dependency $X \rightarrow Y$ **holds**, we mean all the instances in which this dependency is valid or true. Therefore, the sentence *"If $X \rightarrow Y$ holds, then $XZ \rightarrow YZ$ holds"* can be rephrased as *"In any instance in which $X \rightarrow Y$ is valid, the dependency $XZ \rightarrow YZ$ is valid as well".*

The Armstrong axioms allow us to define the closure of a set of dependencies as the iterative application of these axioms over a set of dependencies.

Definition 2.2. *Σ^+ denotes the closure of a set of dependencies Σ, and can be constructed thanks to the iterative application of the Armstrong axioms over Σ. This iterative application terminates when no new dependency can be added, and it is finite. Therefore, Σ^+ contains the largest set of dependencies that hold in all instances in which all the dependencies in Σ hold.*

The closure of a set of dependencies induces the definition of the cover of such a set of dependencies.

Definition 2.3. *The **cover** or **basis** of a set of dependencies Σ is any set Σ' such that $\Sigma'^+ = \Sigma^+$.*

3. Covers of Dependencies

We now present the different types of covers that are present and adopted in the three fields, namely RDBM, FCA, and Logic. Since the definition of a cover (Definition 2.3) is very generic, the covers reviewed here have been defined with respect to specific characteristics and for different purposes.

3.1. Four Main Characteristics of Covers

In general terms, a cover is simply a set of dependencies Σ. The definition of a cover is very generic and below we introduce some relevant properties which are useful for characterizing the different covers. In particular, Σ is equivalent to another set of dependencies Σ' modulo the Armstrong axioms when the closures Σ^+ and Σ'^+ are the same.

There are three sources of redundancy in a set of dependencies: reflexivity, augmentation and transitivity (with respect to the Armstrong axioms 2.1), but here dependencies that can be derived by reflexivity 2.1 are **trivial** and not considered in the following discussion. This assumption does not invalidate any of the arguments that are to be presented, but simplifies the discussion. We will present three bases that try to reduce the number of dependencies that are needed in order to compute the closure $\mathrm{closure}_\Sigma$ for any set of attributes. But first, we need to characterize the said bases according to the different ways to remove redundancy that are used.

Definition 3.1. *A set of dependencies Σ is **left-reduced** if and only if for all $X \to Y \in \Sigma$ there is no $X' \to Y$, where $X' \subset X$, such that changing $X \to Y$ by $X' \to Y$ in Σ gives an equivalent base.*

The process of left-reducing a set of dependencies is also mentioned as the removal of **extraneous attributes** in the left-hand sides of all dependencies. As it can be expected, an attribute x is extraneous in the left-hand side of a dependency $\sigma \in \Sigma$ if removing x from the left-hand side in σ does not change Σ^+.

Example 3.1. *Let us take the set of dependencies $\Sigma = \{\, a \to b, b \to ac, a \to c, bc \to a \,\}$. In this set, the dependency $bc \to a$ contains the extraneous attribute c in the left-hand side, because changing $bc \to a$ by $b \to a$, we have that Σ^+ is the same.*

The equivalent case, in the right-hand side, is right-reduction. An attribute y is extraneous in the right-hand side of a dependency $\sigma \in \Sigma$ if removing y from the right-hand side in σ does not change Σ^+.

Definition 3.2. *A set of dependencies Σ is **non redundant** if and only if $(\Sigma \setminus \sigma)^+ \neq \Sigma^+$ for all $\sigma \in \Sigma$.*

If a set of dependencies Σ is non redundant and all the right-hand sides are singletons then Σ is **right-reduced** (Definition 5.6 in [13]).

Since $\{\, X \to yz \,\} \equiv \{\, X \to y, X \to z \,\}$, there is no difference between considering a cover such that all the right-hand sides are singletons, or just *joining* in one single dependency all those dependencies with the same left-hand side. Actually, when the right-hand sides are singletons, the number of dependencies is “artificially” increased.

Definition 3.3. *A set of dependencies Σ is **direct** if and only if* $\mathrm{closure}_\Sigma(X)$ *can be computed with a single pass of* Σ*, for all* $X \subseteq \mathcal{U}$*.*

As we stated in Section 1, we assume that the computation of $\mathrm{closure}_\Sigma$ is performed by the algorithm **Closure** or **LinClosure**. The Definition 3.3 means that computing the closure of any set of attributes $X \subseteq \mathcal{U}$ implies only one complete exploration of Σ. Actually, this unique exploration represents a very important property, especially when Σ is large or very large. Moreover, it should also be noticed that (i) a cover Σ is direct regardless of how it is represented or sorted, and also (ii) Σ is direct for all possible sets of attributes $X \subseteq \mathcal{U}$.

Definition 3.4. *A set of dependencies Σ is **minimal** if and only if* $|\Sigma| \leq |\Sigma'|$ *for all* Σ' *such that* $\Sigma^+ = \Sigma'^+$.

It is important to notice that when a cover is minimal, it is also non redundant, but the opposite does not necessarily hold. Moreover, let us recall also the importance of computing a non redundant cover given a set of dependencies Σ, and that computing a non redundant cover is equivalent to the logical implication problem.

In the next sections we review three different and major types of covers that are of interest in the RDBM, in Logic, and in FCA.

3.2. The Canonical-Direct Basis

The Canonical-Direct Basis is defined with different characterizations in [12] and aims to remove the redundancy caused by augmentation. The computation of this cover is performed in three steps:

1. All the dependencies in Σ must have one single attribute in the right-hand side, as it was already the case above for the computation of the minimal cover. Again this is performed by simply replacing a dependency $X \to Y$ by the dependencies $X \to y_i$ for all $y_i \in Y$.
2. Σ is closed by "pseudo-transitivity", that is, Augmentation plus transitivity. Then dependencies $X \to y$ such that $y \in X$ are removed.
3. Σ is left-reduced (see Definition 3.1 and step 2 in the construction of the minimal cover).

It can be noticed that there is no removal of redundant dependencies. The only source of redundancy that is taken into account and removed is the one generated by the application of augmentation, but not of transitivity. The Canonical-Direct Basis is not necessarily minimal, nor it is non-redundant, but it is direct.

Example 3.2. *We continue with this example:* $\Sigma = \{\, a \to b, b \to ac, a \to c, bc \to a \,\}$*. Applying step 1 produces* $\Sigma = \{\, a \to b, b \to a, b \to c, a \to c, bc \to a \,\}$*. Here, step 2 consists in closing* Σ *by pseudo-transitivity, which outputs:* $\Sigma' = \{\, a \to b, b \to a, b \to c, a \to c, bc \to a, ac \to a, ab \to a, ab \to b, ac \to c \,\}$*. When applying step 2 to left-reduce* Σ*, and since* $bc \to a$ *can be left-reduced to* $b \to a$*, the following equivalent set is obtained and constitutes the final result:* $\Sigma''' = \{\, a \to b, b \to a, b \to c, a \to c \,\}$*, Since there is no removal of redundant dependencies, then dependency* $a \to c$ *appears in this canonical-direct base.*

We need to notice that this base is also characterized in Formal Concept Analysis by the so called **minimal generators** (or minimal generating set). A minimal generator is defined in [9] (Section 2.3.3) as follows: *A generating set of a closed set A is a subset $S \subseteq A$ such that $A = S''$, and, obviously, a minimal generating set of A is a subset of A minimal with respect to this property.* A similar definition exists in [18]: *A minimal generator B of a closed set F is also called a base for F , i.e. $\phi(B) = F$ and $\phi(A) \subset \phi(B)$ for every $A \subset B$, or a free subset, i.e. for every $x \in B$, $x \in \phi(B \setminus x)$* (where ϕ is a closure operator). What is also relevant is that, in this same paper, the characterization of the canonical direct base Σ_{cdb} is defined as follows: $\Sigma_{cdb} = \{\, B \to \mathrm{closure}_\Sigma(B) \setminus B : B \subseteq \mathcal{U}$ is a minimal generator of $\mathrm{closure}_\Sigma(B)\,\}$. That is, the left-hand sides of a Canonical-Direct Basis are the set of minimal generators of the implicit closure operator.

3.3. The Minimal Cover in the RDBM and its variations

Below we introduce the **Minimal Cover**, which is very popular among the RDBM community and can be found in most of the database textbooks under different names (see Table 1). This cover aims to remove the redundancy caused by both augmentation and transitivity.

Name	Ref	Where
Canonical Cover	Maier [13]	p. 79, Section 5.6
Minimal Cover	Ullman [3]	p. 390
Minimal Cover	Abiteboul [14]	p. 286, Exercice 11.16
Irreducible Set of Dependencies	Date [19]	p. 341, Section 11.6
Minimal Cover	Elmasri [20]	p. 549, Section 16.1.3
Canonical Cover	Silberschatz [21]	p. 324, Section 7.4.3

Table 1
References to the Minimal Cover in RDBM textbooks.

The computation of the Minimal Cover is performed in three steps:

1. All the dependencies in Σ must have only one single attribute in the right-hand side. This is performed by simply replacing a dependency $X \to Y$ by the dependencies $X \to y_i$ for all $y_i \in Y$.
2. Σ is left-reduced. This is performed by changing a dependency $X \to y$ by a dependency $X' \to y$, where $X' \subset X$, whenever $(\Sigma \setminus \{X \to Y\} \cup \{X' \to Y\})^+ \equiv \Sigma^+$ (see Definition 3.1).
3. Redundant dependencies are removed from Σ (see Definition 3.2).

It is important to notice that the order of steps 2 and 3 is relevant and mandatory. Section 5.3 in [13] includes a discussion explaining why left-reduction needs to be performed before the removal of redundant dependencies. The output of computing the Minimal Cover depends also on the order in which dependencies are processed. As a consequence it comes that there may be different minimal covers for the same Σ. Finally, the Minimal Cover does not ensure directness.

Example 3.3 (Adapted from Section 5.2 in [13]). *Let us suppose that we have the following set of dependencies:* $\Sigma = \{ a \to b, b \to ac, a \to c, bc \to a \}$. *Applying step 1 outputs* $\Sigma = \{ a \to b, b \to a, b \to c, a \to c, bc \to a \}$. *Then step 2 is applied to left-reduce* Σ. *Since* $bc \to a$ *can be left-reduced to* $b \to a$ *(thanks to Augmentation), then the following equivalent set is produced:* $\Sigma' = \{ a \to b, b \to a, b \to c, a \to c \}$. *Finally, applying step 3 outputs:* $\Sigma'' = \{ a \to bc, b \to a \}$. *By contrast, let us assume that the order of* Σ *is changed as follows:* $\Sigma = \{ a \to b, a \to c, b \to ac, bc \to a \}$. *Applying step 1 yields the same set as above:* $\Sigma = \{ a \to b, a \to c, b \to a, b \to c, bc \to a \}$. *When applying step 2, it comes:* $\Sigma' = \{ a \to b, a \to c, b \to a, b \to c \}$. *And finally applying step 3 outputs the base* $\Sigma' = \{ a \to b, b \to ac \}$.

3.4. The Duquenne-Guigues basis

The **Duquenne-Guigues basis** [17, 8], also called the *Canonical Basis* in the FCA community, is the cover based on pseudo-closed sets [8]. More precisely, it is defined as follows:

Definition 3.5. *The **Duquenne-Guigues basis** of a set of dependencies* Σ *is defined as*

$$\{ X \to \mathrm{closure}_\Sigma(X) \mid X \subseteq \mathcal{U} \text{ and } X \text{ pseudoclosed} \}$$

where the definition of a pseudo-closed set of attributes w.r.t. a set of dependencies Σ is:

Definition 3.6. *Let* Σ *be a set of dependencies, and* $\mathcal{U}$ *the related set of attributes.* $X \subseteq \mathcal{U}$ *is **pseudoclosed** if:*

1. $X \neq \mathrm{closure}_\Sigma(X)$, *that is,* X *is not closed.*
2. *If* $Y \subset X$ *is a proper subset of* X *and pseudo-closed, then* $\mathrm{closure}_\Sigma(Y) \subseteq X$.

This base is not direct, but it is minimal and non-redundant. According to [12], this base is also presented by Maier in [13], where it is called the *Minimum Cover*: *"It has been obtained independently (and with different formulations) by Maier (Minimum Cover), and Guigues and Duquenne (Duquenne-Guigues basis)"*. We interpret the expression *with different formulations* as the fact that the left and right-hand sides of the Duquenne-Guigues basis can be further left/right-reduced. This is: modulo left/right-reduction, the Minimum Cover and the Duquenne-Guigues basis are essentially the same. We note that [13] (Section 5.6.2) presents a way to compute a Minimum Cover out of a non redundant set of dependencies. However, at present, the use and popularity of the Duquenne-Guigues basis seems to be rather restricted within the FCA community [9, 8].

Example 3.4. *Let us consider Example 3.1:* $\Sigma = \{ a \to b, b \to ac, a \to c, bc \to a \}$.

As in Σ *the pseudo-closed sets are simply* $\{ a \}$ *and* $\{ b \}$, *the following Duquenne-Guigues basis is obtained:* $\Sigma' = \{ a \to bc, b \to ac \}$.

3.5. Some Notes on Complexity

Concerning the complexity of the computation of the different bases, in all cases the worst-case scenario is that of a base of exponential size with respect to the number of attributes as well as the computing time of such bases.

In [22] it is proved that the complexity of computing the Canonical-Direct Basis is of order $\mathcal{O}(n^2\ (\frac{m}{2})^2\ 2^m))$, where m is the number of attributes and n is the number of records (the size of the dataset). In [16] it is proved that any algorithm computing the dependencies that hold in a dataset has a complexity of $\mathcal{O}(\binom{n}{\frac{n}{2}})$, where n is the number of attributes. As for the Duquenne-Guigues basis, in [23] is is proved that (a) the size of a Duquenne-Guigues basis is exponential in the worst-case and that (b) to estimate its size without computing it is #P-hard. In fact, deciding whether a set is a pseudo-closed set is a coNP-complete problem [24].

4. Discussion and Conclusion

Relating the three main covers plus the D-Basis is relevant and very interesting. Actually, while the D-Basis and the Canonical-Direct Basis are related by a subset relationship, such a relationship is not known to exist between the Duquenne-Guigues basis and the Minimum Cover, or between the Canonical-Direct Basis and the Duquenne-Guigues basis. In fact, in [12], in the last sentence before the acknowledgements page 28, it is stated that:

> *We conclude that this paper is contradicting a conjecture of the literature (in [37]). Indeed, one observes that the premise of implication (10) of Σ_{cd} (a direct cover) is not contained in a premise of any implication of Σ_{can} (the Duquenne-Guigues base).*

This sentence goes back to a conjecture stated in [25] and can be interpreted as follows. The left-hand sides of a Duquenne-Guigues basis, which is minimal, may not be a subset of the left-hand sides of a direct cover.

The RDBM, Logic, and FCA fields are addressing two different, yet related problems: the implication problem and the computation of a compact and representative set –cover or base– of a complete set of dependencies. Although the first question is solved in the three fields thanks to the same algorithms, that is, Closure or Linclosure, this unanimity disappears in confronting with the choice of a cover. Table 2 summarizes the three types of covers reviewed in Section 3, plus the D-Basis. It can be noticed that the Canonical-Direct Basis and the D-Basis do not keep dependencies that can be inferred by the application of the augmentation axiom: $X \rightarrow Y \models XZ \rightarrow YZ$, while they include dependencies that can be inferred by transitivity. This additional amount of information is enough to make direct these two covers. By contrast, the Minimal Cover does not contain dependencies that can be inferred by augmentation or transitivity as they are removed in the last step of the computation. And this explains why the Minimal Cover is not direct.

We have there a kind of "no free lunch theorem": the more information a base is keeping the more direct the base can be. By contrast, minimality and non redundancy do not favor directness.

	Canonical Minimal	Canonical Direct	Duquenne-Guigues Minimum	D-Basis
(found in)	RDBM	RDBM, Logic	FCA/RDBM	Lattice Th.
Minimum	no	no	yes	no
Direct	no	yes	no	yes
Redundant	no	yes	no	yes
Unique	no	yes	yes	yes

Table 2
Comparing the characteristics of the four bases.

The lack of unanimity can also be noticed in the RDBM only. Indeed textbooks such as [14, 13, 3, 19, 20, 21] tend to present the Minimal Cover as the preferred cover, while algorithms computing FDs output the Canonical-Direct Basis [26]. This can be interpreted as follows: textbooks are preferring a cover left-reduced and non-redundant, and, hence, containing less and more compact information. However, for discovering FDs, a left-reduced cover is computed without removing redundant dependencies. A possible explanation is that algorithms computing FDs take a dataset as input. Then it is easy to perform a left-reduction w.r.t. the input dataset by removing an attribute from the left-hand side and test whether the dependency still holds. However, a redundancy test is different in the sense that it can only be performed w.r.t. a set of dependencies, *once this set has completely been computed.* Such a test is of a different nature as it is not performed w.r.t. a dataset. Moreover, this does not prevent any algorithm from performing it.

Although the Minimum Cover (or Duquenne-Guigues basis) is also introduced in the RDBM field, it has not enjoyed the same popularity as the Minimal Cover. In fact, the Minimum Cover is introduced and discussed only in Maier [13]. This lack of popularity is probably due to the rather intricate characterization of the Minimum Cover and the related algorithm at that time (see for example Section 5.6, Chapter 5 in [13]). This is especially true when compared with the simplicity and expressiveness of the presentation of the Minimal Cover. The characterization of Minimum Cover cannot compete either with the clear characterization of the Duquenne-Guigues basis in FCA, or the simplicity of the NextClosure Algorithm in [9]. However, this "simplicity" is not for free and it comes at the expense of the whole theoretical framework of FCA.

The choice of a cover can be decided depending on the performance that Closure or Linclosure are offering. Both Closure and Linclosure are closely dependent on the "nature" of the set of dependencies Σ. By "nature" we mean the different characteristics explained in Section 3, which have an impact on the amount of information as well as on the number of dependencies considered. If Σ is a direct cover (Definition 3.3), both algorithms have to perform only a single pass of their outer loop. If the cover is not direct, e.g., Canonical-Direct Basis or Duquenne-Guigues basis, then, the number of iterations of the outer loop may be larger, while, at the same time, the number of iterations of the inner loop may be shorter. Again, we are facing the well-known trade-off between "expressivity and complexity": the more expressive in terms of information containment a cover is, the higher the cost of Closure and Linclosure is.

Constructing decision quivers

Egor Dudyrev[1,2,*,†], Sergei O. Kuznetsov[1,†] and Amedeo Napoli[2,†]

[1]*HSE University, 20 Myasnitskaya St, Moscow, 101000, Russian Federation*

[2]*Université de Lorraine, CNRS, LORIA, F-54000 Nancy, France*

Abstract

Rule Learning and Formal Concept Analysis (FCA) are two fields of science that study similar topic yet speak in a very different terms. This paper describes rule-based machine learning models with FCA-based terminology which results in decision quiver model. A decision quiver, discussed in the paper, is a supervised machine learning model that is based on intents, generators of intents, and predictions for each intent (or generator). We show that the finding of the optimal set of intents is a cornerstone task in constructing a decision quiver (and thus, any rule-based model). The paper finishes with the baseline algorithm to construct decision quivers. The algorithm produces machine learning models that are much smaller than the state-of-the-art ensembles of decision trees, yet that offer the similar quality of predictions.

Keywords

Supervised Machine Learning, Explainable Artificial Intelligence, Formal Concept Analysis

1. Introduction

Rule Learning [1] and Formal Concept Analysis (FCA) [2] are two fields of science that study similar topic yet speak in a very different terms. Rule Learning searches for rules that could accurately predict the attributes of unseen data. While FCA focuses on studying dependencies in only the given data. This paper attempts to combine the language of FCA and the goal of Rule Learning in one model called Decision Quivers. Thus, we combine the mathematicity of FCA with the applicability of Rule learning.

Rule learning is a mature and well-recognised research area mainly concerned with finding the Boolean rules (i.e. ”rules”) from the given attributes that are able to predict the value of “target” attribute. One specific configuration of a rule-based model – an ensemble of decision trees – is considered among the state-of-the-art machine learning models on tabular data. However, the last big increment in the prediction quality of the ensembles was introduced in the year 2016 [3] and is based on ensembling decision trees that are known since (at least) the year 1986 [4]. The later studies on Rule learning focus on interpreting and explaining big rule-based models, ensembles of decision trees especially [5] [6] [7] [8]. We attribute the lack of novel

Submitted for the 11th International Workshop ”What can FCA do for Artificial Intelligence?”, FCA4AI 2023, co-located with IJCAI 2023, August 20 2023, Macao, S.A.R. China

*Corresponding author.

†These authors contributed equally.

eo.dudyrev@hse.ru (E. Dudyrev); skuznetsov@hse.ru (S. O. Kuznetsov); amedeo.napoli@loria.fr (A. Napoli)
0000-0002-2144-3308 (E. Dudyrev); 0000-0003-3284-9001 (S. O. Kuznetsov); 0000-0001-5236-9561 (A. Napoli)

state-of-the-art models in the area to the lack of a good formalism to describe rule-based models. Such formalism could also propose new ways to interpret and explain big rule-based models.

Formal Concept Analysis is a formalism aimed at analysing data based on discrete (often binary) descriptions. The FCA focus on binary descriptions promises its good applicability to become a formalism for rule-based models.

The natural connections between FCA and rule-based machine learning were covered in many works: [9] [10] [11] [12] [13] [14]. This paper does not attempt to highlight any new connection between FCA and rule-based models. Instead, we discuss a model for rule-based machine learning called Decision Quiver. The model is defined using the very basic notions of FCA – closed descriptions and generators. The notion of generators allows decision quiver describe any other rule-based model. While the notion of closed descriptions greatly shrinks the search space while constructing the model.

The paper is structured as follows. Section 2 recalls the basics of Formal Concept Analysis and Supervised Machine Learning, and defines Decision Quivers. Section 3 introduces the pipeline and the simple algorithm to construct decision quivers. Section 4 evaluates the algorithm on some of LUCS-KDD datasets. Section 5 concludes the paper.

2. Background

This subsection introduces definitions we use throughout the paper. Firstly, we provide the basic terms of Formal Concept Analysis to describe the rule models. Secondly, we describe the space of premises that contains the machine learning models discussed in the paper. Thirdly, we describe the main topics of a binary classification in the language in the FCA notation.

2.1. Formal Concept Analysis

In Formal Concept Analysis the data is represented as a formal context $K = (G, M, I)$, which is a triple of a set of objects G, a set of attributes M and the relation $I \subseteq G \times M$ among them.

A running example of a formal context is provided in Table 1 (together with "Target $\mathbb{Y}$" column and "test objects" rows that will be introduced in the following sections). To lighten the notation, we will represent a subset of attributes from the running example as a concatenation of these attributes: e.g. we denote the subset of attributes $\{c, h\}$ as ch.

We call any subset of attributes $D \subseteq M$ a description. And we denote the set of all descriptions (i.e. the powerset of M) as 2^M:

$$2^M = \{D \mid \forall D \subseteq M\} \tag{1}$$

Now we can define two "prime" operations: A' would describe all attributes M shared by objects $A \subseteq G$, and B' would describe all objects G covered by attributes $B \subseteq M$. Prime operations:

$$\begin{aligned} A' &= \{m \in M \mid \forall g \in A : (g, m) \in I\}, \quad A \subseteq G \\ B' &= \{g \in G \mid \forall m \in B : (g, m) \in I\}, \quad B \subseteq M \end{aligned} \tag{2}$$

Two prime operations, combined together, result in "double prime" operator $(\cdot)''$ that has the properties of a closure operator. For example, if B is a subset of attributes M then B'' is the closure of B on set M.

		Attributes M				*Target $\mathbb{Y}$*
		l	w	c	h	τ
	dog	x		x	x	100%
Objects	corn	x				0%
G	bream		x	x	x	0%
	egg					0%
Test	reed	x	x			? (0%)
objects	frog	x	x	x	x	? (0%)
Attr. names		lives on land	lives in water	can move	has limbs	is mammal

Table 1
Running example of a formal context with additional target column $\mathbb{Y}$ and test objects

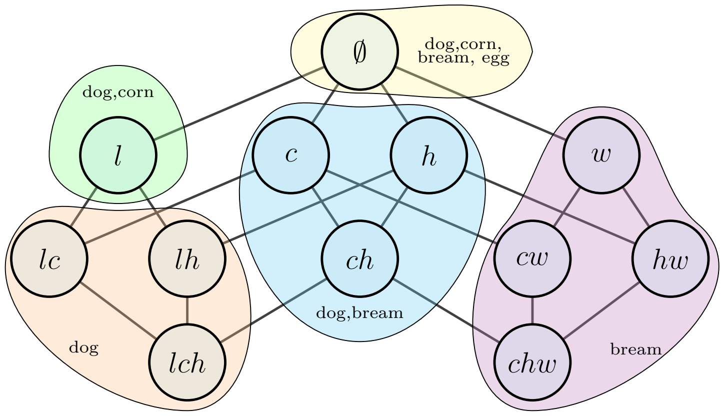


Figure 1: Descriptions of the example context grouped by the subsets of objects they describe. Descriptions of the empty set of objects are omitted.

In FCA terminology, a closed description $B \subseteq M, B'' = B$ is also referred to as an *intent*. The set of all intent is denoted by $\mathbb{B}$ and, together with set inclusion order, forms a lattice :

$$\mathbb{B} = \{B \subseteq M \mid B'' = B\} \tag{3}$$

Two different subsets of attributes $D, E \subseteq M, D \neq E$ can describe the same set of objects $D' = E'$ and, consequently, have the same closure $D'' = E''$. Such descriptions are called *equivalent*. Equivalence class $[D]$ of description $D \subseteq M$ is denoted as $[D]$:

$$[D] = \{E \subseteq M \mid E'' = D''\}. \tag{4}$$

Line diagram on Figure 1 shows the descriptions of a context from the running from Table 1 grouped by the equivalence classes. Descriptions equivalent to M (that describe no objects) are omitted as they would make the diagram harder to read.

In each equivalence class $[D], D \subseteq M$, there is a single maximal description $D'' = B$ (also called closed description or intent) and possibly many minimal descriptions, that are called "keys". Let $B \subseteq M$ be a closed description $B'' = B$, then $keys(B)$ is the set of minimal descriptions, equivalent to B:

$$keys(B) = \{E \in [B] \mid \nexists D \in [B] \, : \, D \subset E\}, \quad B \subseteq M \tag{5}$$

2.2. Supervised Machine Learning

This subsection covers the basic ideas of Supervised Machine Learning and introduces our notation for these terms.

Let us define a formal context (G, M, I), a space of target values $\mathbb{Y}$, and a target label $\tau(g) \in \mathbb{Y}$ for each object $g \in G$. The task of supervised machine learning is to find a function ψ that maps any description $X \subseteq M$ to a target value $\psi(X) \in \mathbb{Y}$ so that, for any object $g \in G$, the prediction $\psi(g')$ would be close to the target label $\tau(g)$. The "closeness" of target labels τ and predictions ψ on objects G is evaluated by non-negative loss function $\mathscr{L}(\tau, \psi \mid G) \in \mathbb{R}_+$, where $\mathscr{L}(\tau, \psi \mid G) = 0$ means that ψ is optimal. Here we provide two loss functions that can be used to evaluate the prediction function ψ: negative F1-score $\mathscr{L}_{F1}$ for binary classification task (i.e. when $\mathbb{Y} = \{0, 1\}$), and Mean Squared Error (MSE) $\mathscr{L}_{MSE}$ for regression task (i.e. when $\mathbb{Y} = \mathbb{R}$).

$$\mathscr{L}_{F1}(\tau, \psi \mid G) = 1 - 2\frac{\sum_{g \in G} \tau(g)\psi(g')}{\sum_{g \in G} \big(1 - \tau(g)\psi(g')\big)}$$

$$\mathscr{L}_{MSE}(\tau, \psi \mid G) = \frac{1}{|G|} \sum_{g \in G} \big(\tau(g) - \psi(g')\big)^2$$

It should be noted that, since the loss function $\mathscr{L}$ is evaluated on objects G, the "best" prediction function ψ for these objects would be the function $\psi \, : \, g' \mapsto \tau(g)$ that predicts the label $\tau(g)$ of every objects $g \in G$ based on its description g'. Such function ψ would give zero loss on objects G but it will not extrapolate well on new descriptions, not shared by objects G. This issue is often mitigated by introducing the "test" formal context $(G_{\text{test}}, M, I_{\text{test}})$ with target labels $\tau_{\text{test}} \, : \, G_{\text{test}} \to \mathbb{Y}$. Then, prediction function ψ is constructed by minimizing the loss on "train" context (G, M, I), but the final evaluation of the loss of function ψ is done on the test context $(G_{\text{test}}, M, I_{\text{test}})$. In the running example from Table 1, objects G_{test} are denoted as *Test objects*.

In what follows we use the "average" operation over the target space (Equation 6). We assume, therefore, that the space $\mathbb{Y}$ is "averageble", i.e. for every list of tuple of target values Y, its average $\text{avg}(Y)$ is also a target value from $\mathbb{Y}$. This assumption does not make the following reasoning too specific, as many Supervised Machine Learning problems can be reformulated to satisfy it. For example, binary classification task suggests only two target values $\mathbb{Y}_{\text{bin}} = \{0, 1\}$. However, it can be reformulated as a regression task with $\mathbb{Y} = [0, 1]$ where the values of $\mathbb{Y}$ represent the probability of an object to belong to the positive class $1 \in \mathbb{Y}_{\text{bin}}$.

$$\text{avg}(Y) = \frac{1}{|Y|} \sum_{y \in Y} y, \quad Y \in \mathbb{Y}^1 \cup \mathbb{Y}^2 \cup \ldots \cup \mathbb{Y}^\infty \tag{6}$$

2.3. Rule set

Given a description $X \subseteq M$, a rule-based machine learning model makes prediction $\psi(X)$ based on a set of rules of the form: "if X is described by $P \subseteq M$ then predict $\varrho(P) \in \mathbb{Y}$". That is, every rule is characterised by a pair $(P, \varrho(P))$ where $P \subseteq M$ is a subset of attributes called *premise*, and $\varrho(P) \in \mathbb{Y}$ is a *prediction* of premise P. A *rule set* is a pair $(\mathscr{P}, \varrho)$ where $\mathscr{P} \subseteq 2^M$ is a set of premises and ϱ is a function that maps each premise to a target value ($\varrho : \mathscr{P} \mapsto \mathbb{Y}$).

Now, let us consider two special cases when predicting with rule set $(\mathscr{P}, \varrho)$. Let $X \subseteq M$ be a description and let $P_1, P_2 \in \mathscr{P}$ be two comparable premises covered by X: $P_1 \subset P_2 \subseteq X$. Then we follow the intuition that a more precise premise P_2 should give a more precise prediction $\varrho(P_2)$, therefore we only use premise P_2 to make a prediction about the target of X. Now, let $P_1, P_2 \in \mathscr{P}$ be incomparable premises, covered by X: $P_1 \not\subseteq P_2, P_2 \not\subseteq P_1, P_1 \subseteq X, P_2 \subseteq X$. Then we use both premises P_1, P_2 to make a prediction for X by averaging their predictions $\psi(X) = \text{avg}\big((\varrho(P_1), \varrho(P_2))\big)$.

With these ideas in mind we define prediction function $\psi_{(\mathscr{P},\varrho)} : 2^M \to \mathbb{Y}$ for rule set $(\mathscr{P}, \varrho)$ as follows:

$$\psi_{(\mathscr{P},\varrho)}(X) = \text{avg}\big((\varrho(P) \mid P \in \mathscr{P}_{X,\max})\big), \quad X \subseteq M \tag{7}$$

$$\text{where } \mathscr{P}_{X,\max} = \{P \in \mathscr{P} \mid P \subseteq X, \forall P_2 \in \mathscr{P} : (P \subset P_2) \implies (P_2 \not\subseteq X)\} \tag{8}$$

Prediction $\varrho(P)$ for a premise $P \in \mathscr{P}$ is often computed as the average of target labels of objects described by P:

$$\varrho(P) = \text{avg}\big((\tau(g) \mid g \in P')\big) \tag{9}$$

Example 1. *Let us provide an example of a rule set $(\mathscr{P}, \varrho)$ constructed on the example context 1. For the sake of readability, we represent the rule set as a set of implications $\{P \implies \varrho(P) \mid P \in \mathscr{P}\}$:*

$$\{\emptyset \implies 25\%,\ l \implies 50\%,\ lc \implies 100\%,\ lh \implies 100\%,\ w \implies 0\%\}. \tag{10}$$

Let us make a prediction for object reed *with description $X = lw$. There are three premises from $\mathscr{P}$ that can be applied for X: $\{\emptyset, l, w\}$. Out of these three, there are two maximal descriptions: $\mathscr{P}_{X,\max} = \{l, w\}$. The former predicts that* reed *is mammal with $\varrho(l) = 50\%$ probability, and the latter predicts that* reed *is mammal with $\varrho(w) = 0\%$ probability. Therefore, the final prediction is $\psi_{(\mathscr{P},\varrho)}(X) = 25\%$ probability of* reed *being a mammal.*

Analogously, let us predict whether object frog *with description $X = lwch$ is a mammal. All premises of $\mathscr{P}$ can be applied for the given X. However, only three of them are maximal: $\mathscr{P}_{X,\max} = \{lc, lh, w\}$. The corresponding premise predictions are $\varrho(lc) = 100\%, \varrho(lh) = 100\%, \varrho(w) = 0\%$. Therefore, the final prediction is $\psi_{(\mathscr{P},\varrho)}(X) = 67\%$ probability of* frog *being a mammal.*

This process of making predictions is schematically depicted in Figure 2.

$$X = lw \xrightarrow{\mathcal{P}_{X,\max}} \{l, w\} \xrightarrow{\varrho} (50\%, 0\%) \xrightarrow{\text{avg.}} 25\% = \psi_{(\mathcal{P},\varrho)}(X)$$

$$X = lwch \xrightarrow{\mathcal{P}_{X,\max}} \{lc, lh, w\} \xrightarrow{\varrho} (100\%, 100\%, 0\%) \xrightarrow{\text{avg.}} 67\% = \psi_{(\mathcal{P},\varrho)}(X)$$

Figure 2: The pipeline of making predictions with Rule Set of Example 1 for a reed ($X = lw$) (top subfigure) and a frog ($X = lwch$) (bottom subfigure)

2.4. Implicitly equivalent premises

Let us rewrite the predictions from Example 1 in more details, considering what objects from G we use to make predictions.

For object *reed* with description $X = lw$ the set $\mathscr{P}$ contains two maximal premises describing X: $\mathscr{P}_{X,\max} = \{l, w\}$. Premise l describes training objects $l' = \{dog, corn\} \subset G$ with corresponding target labels $\tau(dog) = 100\%, \tau(corn) = 0\%$ whose average label is 50%. Premise w describes training object $w' = \{bream\} \subset G$ with target label $\tau(bream) = 0\%$. So the rule set $(\mathscr{P}, \varrho)$ from Example 1 predicts that *reed* is mammal with 25% probability.

For object *frog* with description $X = lwch$ the set $\mathscr{P}$ contains three maximal premises describing X: $\mathscr{P}_{X,\max} = \{lc, lh, w\}$. Premise lc describes training object $l' = \{dog\} \subset G$ with target label $\tau(dog) = 100\%$; premise lh describes the same training object training object $w' = l' = \{dog\} \subset G$ with target label $\tau(dog) = 100\%$; and premise w describes training object $w' = \{bream\} \subset G$ with target label $\tau(bream) = 0\%$. So the rule set $(\mathscr{P}, \varrho)$ from Example 1 predicts that *frog* is mammal with 67% probability.

This process of making predictions is schematically depicted in Figure 3.

$$\begin{tikzcd}
 & & \{dog, corn\} \arrow[r, "\tau"] & (100\%, 0\%) \arrow[r, "\text{avg.}"] & 50\% \arrow[dr, "\text{avg.}"] & \\
X = lw \arrow[r, "\mathcal{P}_{X,\max}"] & \{l, w\} \arrow[ur, "l'"] \arrow[dr, "w'"'] & & & & 25\% = \psi_{(\mathcal{P},\varrho)}(X) \\
 & & \{bream\} \arrow[r, "\tau"] & (0\%) \arrow[r, "\text{avg.}"] & 0\% \arrow[ur, "\text{avg.}"'] &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & \{dog\} \arrow[r, "\tau"] & (100\%) \arrow[r, "\text{avg.}"] & 100\% \arrow[dr, "\text{avg.}"] & \\
X = lwch \arrow[r, "\mathcal{P}_{X,\max}"] & \{lc, lh, w\} \arrow[ur, "lc'"] \arrow[r, "lh'"] \arrow[dr, "w'"'] & \{dog\} \arrow[r, "\tau"] & (100\%) \arrow[r, "\text{avg.}"] & 100\% \arrow[r, "\text{avg.}"] & 67\% = \psi_{(\mathcal{P},\varrho)}(X) \\
 & & \{bream\} \arrow[r, "\tau"] & (0\%) \arrow[r, "\text{avg.}"] & 0\% \arrow[ur, "\text{avg.}"'] &
\end{tikzcd}$$

Figure 3: The explicit pipeline of making predictions with Rule Set of Example 1 for a reed ($X = lw$) (top subfigure) and a frog ($X = lwch$) (bottom subfigure)

Notice that two premises $\{lc, lh\} \subset \mathscr{P}$ used for target prediction of object *frog* ($X = lwch$) have the same extent: $lc' = lh' = \{dog\}$. Therefore, the information that subset of objects $\{dog\}$ has average target of 100% is used two times inside the rule set. In fact , if we count only one premise with extent $\{dog\}$ dog when making the final prediction $\psi_{(\mathscr{P},\varrho)}$ we will get the value 50%, which is closer to the true value 0% than 67%.

There are many possible ways to overcome this issue. For example, one can construct rule set $(\mathscr{P}, \varrho)$ such that the set $\mathscr{P}$ would only contain premises that are either comparable or contradicting (i.e. $\forall P_1, P_2 \in \mathscr{P} : P_1 \subseteq P_2$ or $P_2 \subseteq P_1$ or $\exists m \in P_1,$ s.t. "not m" $\in P_2$). This is the approach, used by decision trees. One can also construct rule set $(\mathscr{P}, \varrho)$ such that the all premises in $\mathscr{P}$ are closed (i.e. $\mathscr{P} \subseteq \mathbb{B} \subseteq 2^M$). Then, there will be no two premises that describe the same set of objects. This is the approach commonly used in FCA literature.

In this paper we propose another approach by enriching the rule set $(\mathscr{P}, \varrho)$ with the set of closures $\mathscr{B}$ of premises $\mathscr{P}$: $\mathscr{B} = \{P'' \mid P \in \mathscr{P}\}$.

2.5. Decision Quiver

The previous sections introduced equivalent descriptions and covered their importance for making predictions with rule sets. This section presents Decision Quiver: a rule set model enriched with the information about equivalent descriptions.

The model of Decision Quiver was introduced in [15] as a directed multigraph (i.e. a quiver) with intents as nodes, generators as edges, and predictions for each nodes. This paper inherits the name of Decision Quiver but presents its definition in a more set-theoretic way, as we have found the latter to be more concise.

Definition 1. Let (G, M, I) be a (training) context and $\mathbb{B}$ be its set of closed descriptions, a *decision quiver* is a triplet $Q = (\mathscr{P}, \mathscr{B}, \varrho)$ of premises $\mathscr{P} \subseteq 2^M$, their intents $\mathscr{B} \subset \mathbb{B} : \{P'' \mid P \in \mathscr{P}\} = \mathscr{B}$, and predictions $\varrho : \mathscr{B} \to \mathbb{Y}$ for every intent.

Given a description $X \subseteq M$, prediction $\psi_{(\mathscr{P},\mathscr{B},\varrho)}(X)$ is computed as the average of predictions ϱ of maximal intents $\mathscr{B}_{X,\max} \subseteq \mathscr{B}$, whose generators $\mathscr{P}_X \subseteq \mathscr{P}$are covered by X.

$$\psi_{(\mathscr{P},\mathscr{B},\varrho)}(X) = \text{avg}\big((\varrho(B) \mid B \in \mathscr{B}_{X,\max})\big), \tag{11}$$

$$\text{where } \mathscr{B}_{X,\max} = \{B \in \mathscr{B} \mid \exists P \in \mathscr{P}_{X,\max} : B = P'', \nexists P_2 \in \mathscr{P}_{X,\max} : B \subset P_2''\} \tag{12}$$

$$\mathscr{P}_{X,\max} = \{P \in \mathscr{P} \mid P \subseteq X, \forall P_2 \in \mathscr{P} : (P \subseteq P_2) \implies (P_2 \not\subseteq X)\} \tag{13}$$

Example 2. *Let us provide an example of a decision quiver $(\mathscr{P}, \mathscr{B}, \varrho)$ having the same premises as a rule set $(\mathscr{P}, \varrho)$ from Example 1. For the sake of readability, we represent the quiver $(\mathscr{P}, \mathscr{B}, \varrho)$ as two sets of implications: $\{P \implies B \mid P \in \mathscr{P}, B \in \mathscr{B}, P'' = B\}$ and $\{B \implies \varrho(B) \mid B \in \mathscr{B}\}$:*

$$\mathscr{P} \to \mathscr{B} : \{\emptyset \implies \emptyset,\ l \implies l,\ lc \implies lch,\ lh \implies lch,\ w \implies chw\},$$
$$\mathscr{B} \to \mathbb{Y} : \{\emptyset \implies 25\%,\ l \implies 50\%,\ lch \implies 100\%,\ chw \implies 0\%\} = \varrho.$$

Contrary to the rule set $(\mathscr{P}, \varrho)$ from Example 1, the quiver $(\mathscr{P}, \mathscr{B}, \varrho)$ "knows" that premises lc and lh are equivalent as they correspond to the same closure lch. Therefore, when making prediction for a frog with description $X = lwch$, the quiver uses only two subsets of objects: $\{dog\} = lwch'$ with average target equal to 100%, and $\{bream\} = chw'$ with average target equal to 0%. This prediction process is schematically represented on Figure 4.

$$\begin{tikzcd}
 & & lch \arrow[r, "\varrho"] & 100\% \arrow[dr, "\text{avg}"] & \\
X = lwch \arrow[r, "\mathcal{P}_{X,\max}"] & \{lc, lh, w\} \arrow[ur, "lc''"] \arrow[ur, "lh''"'] \arrow[dr, "w''"'] & & & 50\% = \psi_{(\mathcal{P},\mathcal{B},\varrho)}(X) \\
 & & chw \arrow[r, "\varrho"] & 0\% \arrow[ur, "\text{avg}"'] &
\end{tikzcd}$$

Figure 4: The pipeline of making predictions with Decision Quiver based on a Rule Set of Example 1 for a frog ($X = lwch$)

3. Quivers construction pipeline

The previous section introduced target-based and arrow-based decision quivers. Now, let us discuss, how these quivers can be constructed.

3.1. Algorithm to find the optimal quiver

Decision quiver $Q = (\mathscr{P}, \mathscr{B}, \varrho)$ consists of three elements. Note that prediction function $\varrho : \mathscr{B} \to \mathbb{Y}$ depends on intents $\mathscr{B}$ and not the premises $\mathscr{P}$. And considering the training formal context, every premise $P \in \mathscr{P}$ describes the same objects as itc closure $B \in \mathscr{B}, P'' = B$. So, when making predictions on the training context, the choice of a specific subset of premises $\mathscr{P}$ is irrelevant. Therefore, the task of finding the optimal quiver $(\mathscr{P}, \mathscr{B}, \varrho)$ on the training formal context reduces to finding the optimal rule set $(\mathscr{B}, \varrho)$ whose premises are limited to intents.

After finding $(\mathscr{B}, \varrho)$ one should select the optimal set of premises $\mathscr{P}$ to construct a quiver $(\mathscr{P}, \mathscr{B}, \varrho)$ that would be generalisable to the test data. However, since premises with the same intent describe the same objects, they are empirically indistinguishable. Thus, the choice of the premises $\mathscr{P}$ relies on a priori intentions: for example, one may want to make premises as precise as possible (then $\mathscr{P} = \mathscr{B}$), or as general as possible (then $\mathscr{P} = \bigcup_{B \in \mathscr{B}} keys(B)$).

More formally, let $\mathscr{L}$ be a loss function, (G, M, I) be a training context, and $Q_{\text{opt}} = (\mathscr{P}_{\text{opt}}, \mathscr{B}_{\text{opt}}, \varrho_{\text{opt}})$ be the quiver, that achieves the minimal loss $\mathscr{L}$ on the context (G, M, I): $\mathscr{L}(\tau, \varrho_{Q_{\text{opt}}} \mid G) \to 0$. Then, the task of finding such optimal quiver Q_{opt} can be separated into three independent steps:

0. Fix the search space of intents $\mathbb{B}_{\text{search}} \subseteq \mathbb{B}$ of context (G, M, I),
1. Find the optimal rule set $(\mathscr{B}_{\text{opt}}, \varrho_{\text{opt}})$ of intents $\mathscr{B}_{\text{opt}} \subseteq \mathbb{B}_{\text{search}}$ and their predictions $\varrho_{\text{opt}} : \mathscr{B}_{\text{opt}} \to \mathbb{Y}$,
2. Construct the set of premises $\mathscr{P}_{\text{opt}}$ of intents $\mathscr{B}_{\text{opt}}$.

The initial step of the pipeline is fixing the search space $\mathbb{B}_{\text{search}}$. For the sake of simplicity, in this paper we assign the search space $\mathbb{B}_{\text{search}}$ to be the space of all intents $\mathbb{B}$. The ways to minimise the search space is one of the future research directions.

The first and the main step of the algorithm is finding the optimal rule set $(\mathscr{B}_{\text{opt}}, \varrho_{\text{opt}})$ where the choice of premises is limited to intents from the search space: $\mathscr{B}_{\text{opt}} \subseteq \mathbb{B}_{\text{search}}$. Remind that the prediction function ϱ_{opt} is often evaluated the same way for every intent. For example, similar to Equation 9, prediction $\varrho_{\text{opt}}(B)$ for an intent $B \in \mathbb{B}$ is the average target label of objects B' described by intent B. Thus, the search for optimal rule set $(\mathscr{B}_{\text{opt}}, \varrho_{\text{opt}})$ is a search for optimal subset of intents $\mathscr{B}_{\text{opt}} \subseteq \mathbb{B}_{\text{search}}$:

$$\mathscr{B}_{\text{opt}} = \arg \min_{\mathscr{B} \subseteq \mathbb{B}_{\text{search}}} \mathscr{L}(\tau, \psi_{(\mathscr{B}, \varrho)} \mid G) \tag{14}$$

The last step of the pipeline is to construct the set of premises $\mathscr{P}_{\text{opt}}$ of quiver Q_{opt} to generalise the latter to the possible test descriptions. Here we propose two options for the set of generators $\mathscr{P}_{\text{opt}}$: the set of closed descriptions $\mathscr{P}_{\text{closed}} = \mathscr{B}_{\text{opt}}$, and the set of keys $\mathscr{P}_{\text{keys}} = \bigcup_{B \in \mathscr{B}_{\text{opt}}} \text{keys}(B)$.

3.2. Algorithm to find the optimal subset of intents

This paper uses the very basic greedy discrete optimisation algorithm to find the optimal subset of intents (i.e. solving the task from Equation 14).

When finding the optimal subset of intents $\mathscr{B}_{\text{opt}} \subseteq \mathbb{B}_{\text{search}}$ we operate the fixed set training context (G, M, I), the fixed targets $\tau : G \to \mathbb{Y}$, and the fixed way to compute the prediction $\varrho(B)$

for an intent $B \in \mathbb{B}_{\text{search}}$ (see eq. 9). Therefore, for the sake of brevity, we define the training loss $\mathscr{L}_{\text{train}}(\mathscr{B})$ of a subset of intents $\mathscr{B}$ as follows:

$$\mathscr{L}_{\text{train}}(\mathscr{B}) = \mathscr{L}(\tau, \psi_{(\mathscr{B},\varrho)} \mid G). \tag{15}$$

The algorithm (to find $\mathscr{B}_{\text{opt}}$ given $\mathbb{B}_{\text{search}}$):

0. Start with $\mathscr{B}$ containing the top intent $\mathscr{B} = \{\emptyset''\}$,
1. Find the intent $B_{\text{opt}} \in \mathbb{B}_{\text{search}}$ that gives the minimal loss, when added to the current set of intents $\mathscr{B}$:
   $B_{\text{opt}} = \arg\min_{B \in \mathbb{B}_{\text{search}}} \mathscr{L}_{\text{train}}(\mathscr{B} \cup \{B\})$
2. Add intent B_{opt} to the set $\mathscr{B}$,
3. Repeat the steps 1, 2 while the set $\mathscr{B}$ contains less than $\mathscr{B}_{\text{max}} \in \mathbb{N}$ elements and the loss decrease is higher than $\epsilon \in \mathbb{R}_+$:
   repeat until $|\mathscr{B}| \leq \mathscr{B}_{\text{max}}, \quad \text{and } \mathscr{L}_{\text{train}}(\mathscr{B}) - \mathscr{L}_{\text{train}}(\mathscr{B} \cup \{B_{\text{opt}}\}) > \epsilon.$

4. Experiments

The main limitation for the current algorithm is that considers the intents search space $\mathbb{B}_{\text{search}}$ as the set of all intents $\mathbb{B}$. Therefore, we can only apply the algorithm on the datasets where we can compute the intents $\mathbb{B}$. For the experiments we chose nine dataset from LUCS-KDD repository [16] that are discretized versions of real-world datasets from UCI repository. The set of intents $\mathbb{B}$ for every dataset was computed by LCM algorithm implemented in Scikit-mine repository.

The selected datasets give the task of multi-class classification. For example, Iris dataset asks to classify a flower into one of the three types (Setosa, Versicolour, and Virginica) based on its petal and sepal lengths and widths. We used F1 score with weighted averaging as a loss function so that it can be applied to all datasets with no adjustments.

For each dataset we fit a gradient boosting model provided by Sci-Kit learn [17] and XGBooost [3] to get the state-of-the-art prediction quality scores. Then we construct two decision quivers for each dataset: one makes predictions via intents (denoted by ”Quiver, intents”), and the other makes predictions via keys of these intents (denoted as ”Quiver, keys”).

Table 2 compares the test prediction quality of models on the selected datasets. One can see that the Gradient boosting models gave the best scores. However, in some cases, quiver models showed comparable decision quality: e.g. Iris, Congres, Breast, Flare datasets. Also, on this data, we see almost no differences between quivers that make predict with intents and quivers that prediction with keys. The only dataset where the difference occurs is Zoo dataset. Currently, we cannot explain this fact as we assumed the difference would be much more apparent. So we should proceed with more profound studies comparing intents and keys as means for predictions.

Now, let us compare the sizes of the models, provided in Table 3. One can see that quiver models contain no more that 6 intents for all the datasets. While each gradient boosting consists of one hundred of decision trees. Note that each decision tree in each gradient boosting consist of many Boolean rules. In that regard, decision quiver model are more efficient, as they are able to achieve the similar prediction quality score with much lesser number of rules.

Table 2
Test F1 score

Dataset	Quiver, intents	Quiver, keys	Grad. Boosting sklearn	XGBoost
zoo	0.819	0.869	1.000	0.929
iris	0.967	0.967	0.967	0.967
ecoli	0.703	0.703	0.757	0.808
congres	0.909	0.909	0.909	0.920
breast	0.936	0.936	0.936	0.936
ticTacToe	0.745	0.745	0.984	0.990
flare	0.878	0.878	0.901	0.880
led7	0.460	0.460	0.774	0.769
nursery	0.863	0.863	0.988	1.000

Table 3
Model complexity in size and in construction time

Dataset	Model size			Construction time (seconds)		
	# intents in quiver	# trees, GB sklearn	# trees, XGBoost	Quiver	GB, sklearn	XGBoost
zoo	6	100	100	12.333	0.159	0.057
iris	3	100	100	2.250	0.067	0.047
ecoli	3	100	100	161.508	0.223	0.125
congres	2	100	100	158.786	0.039	0.037
breast	3	100	100	0.786	0.028	0.032
ticTacToe	5	100	100	53.155	0.060	0.065
flare	1	100	100	661.792	0.204	0.196
led7	5	100	100	2.047	0.961	0.726
nursery	5	100	100	1165.161	2.774	0.835

However, the better algorithm to construct decision quivers should be developed. As the current algorithm construct the quivers for too long. For example, on Nursery dataset, it took XGBoost 0.8 seconds to construct 100 decision trees, and it took decision quivers algorithm 19 minutes to select 5 intents.

5. Conclusion

In this paper we have presented decision quivers as a formalism for describing rule-based machine learning models. We showed that description quiver can describe any rule-based model. And, by incorporating closed descriptions, it can possibly suggest efficient algorithms to create more optimal machine learning models.

We have also presented a general pipeline to construct decision quivers. We showed that the most important part of the pipeline is finding the optimal set of intents. Thus, many FCA-based algorithm can be used to construct decision quivers. The current baseline algorithm for quivers construction is very slow, compared to the state-of-the-art models. However, it can produce much smaller models with the similar prediction quality.

References

[1] J. Fürnkranz, D. Gamberger, N. Lavrač, J. Fürnkranz, D. Gamberger, N. Lavrač, Rule learning in a nutshell, Foundations of Rule Learning (2012) 19–55.

[2] B. Ganter, R. Wille, Formal Concept Analysis, Springer, Berlin, 1999.

[3] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[4] J. R. Quinlan, Induction of decision trees, Machine learning 1 (1986) 81–106.

[5] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence 1 (2019) 206–215. doi:10.1038/s42256-019-0048-x.

[6] C. Bénard, G. Biau, S. Da Veiga, E. Scornet, Interpretable random forests via rule extraction, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2021, pp. 937–945.

[7] H. Lakkaraju, S. H. Bach, J. Leskovec, Interpretable decision sets: A joint framework for description and prediction, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1675–1684.

[8] E. Dudyrev, I. Semenkov, S. O. Kuznetsov, G. Gusev, A. Sharp, O. S. Pianykh, Human knowledge models: Learning applied knowledge from the data, Plos one 17 (2022) e0275814.

[9] S. O. Kuznetsov, Machine learning and formal concept analysis, in: Concept Lattices: Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004. Proceedings 2, Springer, 2004, pp. 287–312.

[10] B. Ganter, S. O. Kuznetsov, Hypotheses and version spaces, in: ICCS, volume 2746, Springer, 2003, pp. 83–95.

[11] E. Dudyrev, S. O. Kuznetsov, Decision concept lattice vs. decision trees and random forests, in: Formal Concept Analysis: 16th International Conference, ICFCA 2021, Strasbourg, France, June 29–July 2, 2021, Proceedings 16, Springer, 2021, pp. 252–260.

[12] R. Belohlavek, B. De Baets, J. Outrata, V. Vychodil, Inducing decision trees via concept lattices, International journal of general systems 38 (2009) 455–467.

[13] T. Hanika, J. Hirth, Conceptual views on tree ensemble classifiers, International Journal of Approximate Reasoning (2023) 108930.

[14] Š. Horvát, L. Antoni, O. Krídlo, A. Szabari, S. Krajči, Generalized decision directed acyclic graphs and their connection with formal concept analysis (2022).

[15] E. Dudyrev, S. O. Kuznetsov, A. Napoli, Description quivers for compact representation of concept lattices and ensembles of decision trees, in: D. Dürrschnabel, D. López Rodríguez (Eds.), Formal Concept Analysis, Springer Nature Switzerland, Cham, 2023, pp. 127–142.

[16] F. Coenen, The lucs-kdd discretised/normalised arm and carm data library, 2003. URL: http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN/.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.