Lazy Classification of Underground Forums Messages Using Pattern Structures

Abdulrahim Ghazal[1], Sergei O. Kuznetsov[2]

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

Abstract
Underground forums are monitored platforms where hackers announce attacks and tools to carry on attacks on businesses or organizations. In this paper, we will experiment on assessing the risk of a dataset of these messages, using pattern structures and a lazy classification scheme, with some introduced complexity-reducing elements and natural language analysis techniques. The results show promising application for this method for this problem, and serve as an introductory step for deeper investigation.

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

1. Introduction

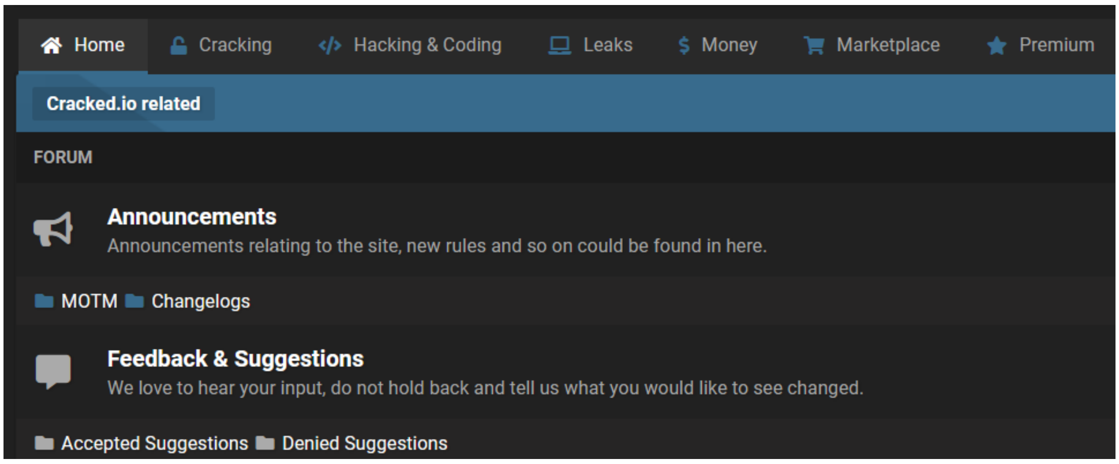
1.1. Threat Intelligence

Threat Intelligence constitutes a very critical part of the cybersecurity world nowadays, with the intensifying cases of cyber attacks that are costing millions of dollars to many industries and governments [1], and improving every day in quantity and quality.

The practice of collecting information about attacks or offers to attack a target from various sources has attracted a lot of research and business attention. This is done by monitoring online forums and instant messaging services, where threat actors post to let other members know that they have some unauthorized access to a network, database or that they made a new tool that can help to achieve the above goals. Most of the serious criminal activity is done for money [2], but sometimes, it has other motives (political statements [3], sabotage or even corporate espionage).

The forums consist of sub-forums and sections that deal with different topics and other administrative sections to control the announcement, sales, and membership processes. In addition to that, many forums have a direct marketplace section, where more strict rules about content are enforced and make these sections designated for sale/buying of illegal products, ranging from cracked software to confidential databases of entities.

Published 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, Proceedings, pp. 63–74.
agazal@hse.ru (A. Ghazal); skuznetsov@hse.ru (S. O. Kuznetsov)



Announcements
Announcements relating to the site, new rules and so on could be found in here.

MOTM Changelogs

Feedback & Suggestions
We love to hear your input, do not hold back and tell us what you would like to see changed.

Accepted Suggestions Denied Suggestions

Figure 1: An example of forum sections (partial structure).

The most prestigious forums also have an "escrow" service that works like a mediator between buyers and sellers on the forum and a type of guarantee of impartial control over the interaction. Many forums also have tiers of membership (VIP, Premium, Golden, etc.). Some forums have no free membership tiers, meaning that one have to pay to access their content. The content posted in paid sections or forums is supposed to have more credibility and be written by more interesting members, depending on the prices, which range from 10$ up to 300$. Most transactions are paid with cryptocurrency.

1.2. Threat Intelligence Workflow

Detecting threats is usually performed with the help of human analysts. The analysts' workflow has three main phases: discovery, reaction and analysis. The most time consuming and difficult to do is the first phase, where the analyst has to browse through thousands of messages posted daily to find credible threats to report. This is even harder when there is a constant stream of spam messages flooding the forums.

This work will focus on helping the human analysts in the discovery phase, using textual analysis of the messages, and learning approaches to help filter out the irrelevant messages, and tag all relevant messages with the right threat category to further ease the next two phases of the workflow.

Now to avoid confusion, we need to define some terms as they are mentioned in the system vs. how they are usually mentioned in the real world. Table 1 gives the terminology.

We need a system that can detect threats without capturing many false positive examples, or missing a true positive case, which carries a high business cost. Another constraint on the needed system is to be time sensitive, as many threats are very volatile, meaning that the threat actor will post about the sale of access for example, in several minutes or hours, some other criminal entity (or in some cases the victims) will contact him about the sale, pay him and request removing the sale announcement. The last important feature of the system is explainability, as human analysts need to understand the reasoning behind a classification result.

Table 1
Terminology Disambiguation.

Usual terminology	Our Terminology	Meaning
Post	Thread	The group of messages that are posted under one topic.
Comment, message, reply	Message	The one item posted in a thread.
-	Last post	The Html tags that contain all the relevant information for extracting a message .
Author, Hacker, original poster	Threat Actor	The person who wrote the message.
Title, Headline	Topic	The thread title.

This will be achieved by building a learning-based system.

The rest of the paper is organized as follows: In Section 2 we recall basic definitions in formal concept analysis and pattern structures. In Section 3 we describe the lazy classification method using pattern structures. Section 4 describes the experimental setting. In Section 5, we discuss the preliminary results of applying the lazy classification to underground forum messages. We conclude the work in section 6.

2. Formal Concept Analysis

2.1. Main Definitions

Formal Concept Analysis (FCA) as defined in [4] is a mathematical theory that is based on concepts and conceptual hierarchy. It is applied for knowledge discovery and data analysis [5, 6].

Let G be a set of objects, M a set of attributes or descriptions of these objects, and $I \subseteq G \times M$ a binary relation between G and M. We call the triple (G,M,I) a formal context. If $g \in G$ has the attribute $m \in M$, then $(g, m) \in I$. We then define the derivation operators $(.)^{'}$ on $A \subseteq G$ and $B \subseteq M$:

$$A^{'} = \{m \in M \mid \forall g \in A : gIm\} \tag{1}$$

$$B^{'} = \{g \in G \mid \forall m \in B : gIm\} \tag{2}$$

We call a pair (A,B) such that $A^{'} = B$ and $B^{'} = A$, a formal concept, and A is called its extent and B is its intent. A partial order $\leq$ is defined on the set of concepts: (A,B) $\leq$ (C,D) iff $A \subseteq C$ and $B \subseteq D$. In this case, (A,B) is called the subconcept and (C,D) a superconcept. This partial order gives rise to a complete lattice on the set of all formal concepts. We call this the concept lattice ζ of the formal context (G,M,I).

The hierarchical structure of concept lattices can be applied at mining association rules [5, 7] ontology design [8, 9], and recommendation systems, because of the ability to explain the rationale behind the recommended item [10].

There is a large focus in FCA domain on building concept lattices and extracting the concepts from a given formal context in an efficient manner, for it to become more applicable in the real world [11]. In [12] researchers compared the performance of several algorithms to build concept lattices and gave insights on which algorithm would be the one to choose depending on the data. There is also a number of works to discuss other applications of FCA in learning [13, 14].

2.2. Pattern Structures

To increase the applications of Formal Concept Analysis, it had to be able to represent more complex data structures, like graphs or non-binary data. This need led to the development of pattern structures [15].

Let G be a set of objects and (D, ⊓) 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 ⊓. Elements of D are ordered by a subsumption relation $\sqsubseteq$ such that a,$b \in D$, then one has $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$. We also define $\delta : G \to D$ as a mapping between objects and their attributes. We call (G, $\underline{D}$, δ) where $\underline{D}$ = (D, ⊓) 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:

$$A^\diamond = \sqcap_{g \in A} \delta(g) \tag{3}$$

$$d^\diamond = \{g \in G \mid d \sqsubseteq \delta(g)\} \tag{4}$$

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, $\underline{D}$, δ).

Pattern structures are helpful in applications with complex data, like graphs, many-valued attributes [5], intervals or interval vectors [6].

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 [16, 17].

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 [16, 17] is described in Algorithm 1.

This algorithm takes $O \mid G \mid \cdot (p(\sqcap + \mid G \mid p(\sqsubseteq)))$ time, where $p(\sqcap)$, $p(\sqsubseteq)$ are times for computing ⊓, $\sqsubseteq$, respectively.

```
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 = (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).
```

3. Experiments

The examples dataset used in the following experiments is composed of underground forum messages as the objects and features extracted from these messages as attributes. After building the concept lattice from this dataset, we construct another dataset for testing the classification. Both datasets contain positive and negative examples. The target attribute is a simple flag stating whether the message is a real threat or not.
The procedure realizing the lazy classification scheme is as follows: we go through the objects of the formal context and check the extension of the intersection of the test object and the concept object, if all objects of this extension have the attribute value, then the test object is classified positively, if all of the objects of the extension do not have the attribute value, the test object is classified negatively, otherwise, the model cannot classify the object. To avoid object selection bias, we shuffle the objects randomly.
Now we move to describe the datasets used in this setting, then talk about the results of the experiments.

3.1. The Datasets

The training examples revolve around messages that are classified as real threats by human analysts. These messages were collected using a system which is a part of a commercial product[1]. The only constraint on these examples is being published on the underground forums in 2021 and on. These examples constitute the positive training dataset part. We checked the underground forums where these messages were written, and the list of these forums is used to generate the negative examples, in a way such that for each positive example X of a forum, Y=n*X negative examples are collected randomly w.r.t the date constraint, where n is a factor, called *balance ratio*, which will be controlled in the experiments. The number of positive training examples is 595. The testing was performed on 960 positive examples and the negative testing examples were constructed in the same manner as training negative examples. These messages are collected from 12 different forums. Several experiments will be run with changing several parameters that might affect the final results. These factors are:

1. Dataset size and distribution: the only constant we have is the number of positive examples used. We test several sizes of negative/positive balance ratio values. In addition to

[1]Provided by the cybersecurity firm Group-IB.

this, we try to make a less random choice of negative examples, by filtering messages that might be spam messages (for instance, messages like “thanks” or “up” are considered to be of low value). We can also choose messages posted only in subsections of forums that are relevant to the classification (if we are trying to find messages about database leaks, we do not need messages coming from the credit cards section). Filtering based on message length is needed also, because some messages are articles, and while they would contain a lot of triggering keywords, they are not actual threats.

2. The intersection operator: we used set-theoretic intersection, then we use interval intersection (We will look into two-sided and one-sided intervals).

3. The number of attributes: the attributes are the values of tf-idf for the words of the messages, and changing the number of words to be in the attributes list is examined, and for which popularity of the keyword we can ignore it and remove it from the list is also considered. The values of attributes in case of theoretic set intersection are binary (the message either have the keyword or not). In case of interval intersection, the value of the attribute would be an interval beginning and ending with the tf-idf value.

4. The tolerance factor α, which represents the probabilistic relaxation allowed for counter examples.

3.2. Assessment Methods

We will have the usual classification assessment methods, but we need to be attentive to the case of unclassified examples. Handling unclassified examples can be done in three different ways:

- Ignoring them by considering them negative examples, and that depends on how harsh our model is about classifying new examples. i.e., if the model classifies almost all positive examples as positive (high recall) and leaves only a very small number of unclassified examples relative to the number of new examples, then it might be possible to classify unclassified examples as negative, as the probability of them being actually positive is very low. This approach is not acceptable in cases where the cost of missing a positive example is high, which is the case for us, as missing a real threat is of a high cost.
- Moving all the unclassified examples to a human analyst, so they can assess their credibility, which can be done only in case the number of unclassified examples is low.
- Removing unclassified examples all together, by having a probabilistic relaxation of the classifier, so that it classifies examples based on how close they are to a class, not as zero-one state.

In addition to the usual definitions of assessment, we define another measure of the improvement this brings to human workers, by how much the messages they have to check shrunk in size. We call it "**saved effort**" and define as $1 - \frac{|G_{uncl}|}{|G_\tau|}$, where G_{uncl} is the set of unclassified examples $G_{uncl} \subseteq G_\tau$.

3.3. Testing Parameters

We first state our parameters that would be used in the future. We have $|G_+| = trp$ positive examples and $|G_-| = trn$ negative examples for the training dataset. We have zero unclassified examples in the training data. The number of negative examples $trn = trp * n$ where n is a positive integer.

The number of attributes is labeled as att. In the test dataset, we have tsp positive examples and tsn examples. The model might have a number of unclassified examples of the test dataset. This set of examples is labeled $uncl$ and it is divided in two subsets: true positive examples that were left unclassified by the model (labeled $pos\text{-}uncl$) and true negative examples that were left unclassified by the model (labeled $neg\text{-}uncl$).

In the method we described above, the explanation of the model is the intersection of attributes of the test object and the concept object that gave us the result of the classification (meaning that the extension of this intersection of attributes all have/do not have the attribute value). This set is denoted by sim.

While we can control the balance of the training dataset in our case, this might not be the real world situation, which means that we would have a varying range of the values of saved effort and F1 measure, presenting us with a trade-off between coverage and model predictions correctness.

4. Results

We performed multiple experiments. The first one would have only binary attributes. The next would have intervals as attributes. In the next two experiments, the data is represented by one-sided intervals. After this, we repeat the pattern structures experiments but with varying values for α. Due to space limitations we present the results at https://github.com/abdulrahimGhazal/FCA-results

4.1. Binary Attributes

The attribute values here are TT-IDF values for the keywords contained in the vectorizer's vocabulary resulting from building the tf-idf model. The value of the attribute in an object would be represented here as:

$$att\_value(keyword) = \begin{cases} 1 & keyword \in vectorizer\ vocab \quad (5) \\ 0 & otherwise \quad (6) \end{cases}$$

We test a range of 5 values of the ratio between the positive to negative examples in the training dataset (from 1 to 5). We also test 5 different values of min_df which is a factor that controls the number of attributes that we would have. It specifies the threshold at which the TT-IDF builder ignores the term if it has less frequency in the documents. Theoretic set intersection is used.

The highest F1 in these experiments occurred on the first experiments where the dataset had the most keywords (lowest min_df step) and the same goes for the saved effort measure. The highest value was F1 = 98.8 and saved effort $=$ 88.8 at (ratio, min_df) $=$ (1, 0.01).

We can observe peaks that happen when we reset the min_df step in the training dataset, then a decline in the F1 afterwards until the next peak. This is a natural observation, since the min_df step increases between the peaks, meaning that we increase the number of ignored keywords that might play a role in classifying the messages correctly.

4.2. Relaxed Binary Attributes

We now repeat the same experiment, using only the lowest balance ratio, varying the values of min_df, and also varying values of α to allow for a certain small amount of counterexamples (objects of the other class matching the intersections used as classifiers).

The highest value was F1 = 98.6 at (α, ratio, min_df) $=$ (90, 1, 0.01) and saved effort $=$ 92.5 at (α, ratio, min_df) $=$ (75, 1, 0.01).

Now we look at using pattern structures to classify underground messages by representing textual data as intervals (two-sided and one-sided).

4.3. Interval Representation

In this next part, we represent the values of the tf-idf as intervals of the floating point value such as if the tf-idf value is x, the attribute value would be [x,x]. In this setting, the intersection operator is defined as an interval that starts with the minimum of the two intervals' starts and ends with the maximum of the two intervals' ends.

$$[a_1, b_1] \sqcap [a_2, b_2] = [min(a_1, a_2), max[b_1, b_2)] \tag{7}$$

The highest F1 in these experiments occurred on the first experiments where the dataset had the most keywords (lowest min_df step) and the same goes for the saved effort measure. The highest value was F1 = 88.0 and saved effort $=$ 87.7 at (ratio, min_df) $=$ (1, 0.01). The values of F1 and saved effort decreases after that with increasing values of min_df.

4.4. One-Sided Interval Representation (Max)

Here, we represent the values of the tf-idf as intervals of the floating point value such as if the tf-idf value is x, the attribute value would be [x,∞].

In this setting, the intersection operator is defined as an interval that starts with the maximum of the two intervals' starts and ends with infinity.

$$[a_1, \infty] \sqcap [a_2, \infty] = [max(a_1, a_2), \infty] \tag{8}$$

The highest F1 in these experiments occurred on the first experiments where the dataset had the most keywords (lowest min_df step) and the same goes for the saved effort measure. The highest value was F1 = 88.0 and saved effort $=$ 87.7 at (ratio, min_df) $=$ (1, 0.01). The values of F1 and saved effort decreases after that with increasing values of min_df.

4.5. One-Sided Interval Representation (Min)

Here, we represent the values of the tf-idf as intervals of the floating point value such as if the tf-idf value is x, the attribute value would be [x,∞].

In this setting, the intersection operator is defined as an interval that starts with the minimum of the two intervals' starts and ends with infinity.

$$[a_1, \infty] \sqcap [a_2, \infty] = [min(a_1, a_2), \infty] \tag{9}$$

The highest F1 in these experiments occurred on the first experiments where the dataset had the most keywords (lowest *min_df* step) and the same goes for the saved effort measure. The highest value was F1 = 89.7 and saved effort = 87.6 at (ratio, *min_df*) = (1, 0.01). The values of F1 and saved effort decreases after that with increasing values of *min_df*.

4.6. Interval Representation With Probabilistic Relaxation

In this next part, we represent the values of the tf-idf as intervals of the floating point value such as if the tf-idf value is x, the attribute value would be [x,x]. As in before, the intersection operator is defined as an interval that starts with the minimum of the two intervals' starts and ends with the maximum of the two intervals' ends.

The difference now is testing several α values (see Algorithm 1).

$$[a_1, b_1] \sqcap [a_2, b_2] = [min(a_1, a_2), max[b_1, b_2)] \tag{10}$$

The highest value was F1 = 94.2 at (α, ratio, *min_df*) = (80, 1, 0.01) and saved effort = 94.1 at (α, ratio, *min_df*) = (75, 1, 0.01). The pattern noticed here is the F1 peaks we get when values of *min_df* increases locally at constant values of α, Then when we increase *min_df* significantly, the values of F1 decreases again.

4.7. One-Sided Interval Representation (Max) With Probabilistic Relaxation

Here, we represent the values of the tf-idf as intervals of the floating point value such as if the tf-idf value is x, the attribute value would be [x,∞]. The difference now is testing several α values (see Algorithm 1).

In this setting, the intersection operator is defined as follows:

$$[a_1, \infty] \sqcap [a_2, \infty] = [max(a_1, a_2), \infty] \tag{11}$$

The highest value was F1 = 91.7 at (α, ratio, *min_df*) = (95, 1, 0.01) and saved effort = 94.5 at (α, ratio, *min_df*) = ((85,90,95), 1, 0.01). The pattern noticed here is the F1 peaks we get when values of *min_df* increases locally at constant values of α, Then when we increase *min_df* significantly, the values of F1 decreases again.

4.8. One-Sided Interval Representation (Min) With Probabilistic Relaxation

Here, we represent the values of the tf-idf as intervals of the floating point value such as if the tf-idf value is x, the attribute value would be [x,∞]. The difference now is testing several α values (see Algorithm 1).

In this setting, the intersection operator is defined as an interval that starts with the minimum of the two intervals' starts and ends with infinity.

$$[a_1, \infty] \sqcap [a_2, \infty] = [min(a_1, a_2), \infty] \tag{12}$$

The highest value was F1 = 94.1 at (α, ratio, min_df) = (75, 1, 0.01) and saved effort = 94.2 at (α, ratio, min_df) = ((75,80), 1, 0.01). The pattern noticed here is the F1 peaks we get when values of min_df increases locally at constant values of α, Then when we increase min_df significantly, the values of F1 decreases again.

4.9. Discussion

Looking at the results, we can see that the binary values of the attributes model performed the best, because the attributes values and intersection method there (the theoretical set intersection) is the most restrictive among all others, giving a little room for error, but the issue with such method is that it suffers from this restrictions when totally new keywords are introduced, as it does not accept any keywords which were not used in the exact intersection.

We can also see that the less restrictive the conditions of the intersection, the less accurate it will be. The best conditions of pattern structures in non-binary representation of the data in our example is the Minimum one-sided intervals, because by its nature, it is the most inclusive of values that are non-zero in the classification scheme.

The next best one is the Maximum one-sided interval. While worse than the Minimum one-sided interval, because it is restricting attribute values to a smaller interval, it still outperforms the interval representation, because of how limited the values of interval representation are.

When relaxation is introduced, we can see that the values of F1 and the saved effort are higher than in the case with no relaxation, supporting that flexibility is useful in case of text messages classification.

The sizes of the *pos-uncl* were always smaller than *neg-uncl*, because of the limited set of keywords the model would have compared to the large amount of possible negative messages' keywords.

5. Conclusion

Underground forum messages are hackers announcements that are shared on the internet about attacks or tools used to carry out attacks. We presented an FCA-based approach for classifying forum messages based on their probability of being risky. The results of experiments show that the use of binary attributes (standard FCA) gave better accuracy, while the use of interval pattern structures gave better saved effort.

6. Acknowledgements

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

References

[1] McAfee, The Economic Impact of Cybercrime: No Slowing Down, Technical Report, Santa Clara, CA, USA, 2021.

[2] S. Pastrana, D. R. Thomas, A. Hutchings, R. Clayton, Crimebb: Enabling cybercrime research on underground forums at scale, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1845–1854.

[3] Zone-H Homepage Team, Zone-h homepage, 2022. URL: http://www.zone-h.org/.

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

[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] S. M. Dias, N. J. Vieira, Concept lattices reduction: Definition, analysis and classification, Expert Systems with Applications 42 (2015) 7084–7097.

[12] S. O. Kuznetsov, S. A. Obiedkov, Comparing performance of algorithms for generating concept lattices, Journal of Experimental & Theoretical Artificial Intelligence 14 (2002) 189–216.

[13] S. O. Kuznetsov, Machine learning and formal concept analysis, in: International Conference on Formal Concept Analysis, Springer, 2004, pp. 287–312.

[14] S. O. Kuznetsov, N. Makhazhanov, M. Ushakov, On neural network architecture based on concept lattices, in: International Symposium on Methodologies for Intelligent Systems, Springer, 2017, pp. 653–663.

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

[16] 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.

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