Intrinsically Interpretable Document Classification via Concept Lattices

Eric George Parakal[1,*], Sergei O. Kuznetsov[1]

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

Abstract

Explanations for the predictions made by Machine Learning (ML) models are best framed in terms of abstract, high-level concepts that are easily comprehensible to human beings. The use of such concepts constitutes a subfield of interpretability methods known as concept-based explanations. This work uses concept-based explanations to build an intrinsically interpretable document classifier using a combination of Formal Concept Analysis (FCA) and approaches from applied graph theory. FCA is used to formalize the vague notion of concepts in terms of the formal concepts found in the concept lattices of various document classes. The graph of the lattice covering relation helps to utilize the topological information present in the document-class concept lattices for classifying documents. Finally, the formal concepts that made the strongest contributions to the predictions of the document classifier are revealed, along with their intents; thereby making their contribution more comprehensible to human beings.

Keywords

Formal Concept Analysis, Explainable Artificial Intelligence, Natural Language Processing

1. Introduction

The extraordinary predictive performance of contemporary Deep Neural Networks (DNNs) for a wide variety of tasks can be largely attributed to their ability to generalize the solving of a task by utilizing a vast number of neuronal parameters. However, the ability to generalize a task also leads to DNNs being more complex with respect to their design, as compared to other ML models. This complexity causes DNNs to be perceived as opaque in terms of their predictive process and consequently, leads to a lack of verifiability with regard to their predictions. Thus, despite their extraordinary predictive performance, DNNs are not adopted for use in high-risk environments such as finance, medicine and the judiciary system due to a lack of trust in their predictions.

This work details the implementation and results obtained from building an intrinsically interpretable document classifier, by utilizing the conceptual hierarchies found in the concept lattices for each document class obtained via FCA. The work as such can then be categorized as belonging to the field of Explainable Artificial Intelligence (XAI).

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. 9–22.

*Corresponding author.

eparakal@hse.ru (E. G. Parakal); skuznetsov@hse.ru (S. O. Kuznetsov)
0000-0002-2059-1608 (E. G. Parakal); 0000-0003-3284-9001 (S. O. Kuznetsov)

With respect to the field of XAI, this work can be further categorized as belonging to the sub-field of concept-based explanations. The fundamental principle of concept-based explanations is that the explanations regarding the predictions of a DNN are best comprehended if they are framed in terms of abstract, high-level concepts. This principle is akin to the process of human reasoning, whereby concepts are informally related to groupings of examples according to the similarity of their descriptions.

Earlier works concerning concept-based explanations can be categorized as post hoc interpretability methods, meaning that they explain the predictions of a DNN after it has finished training. However, more recent concept-based explanation methods belong to the category of intrinsic interpretability methods, wherein a DNN is interpretable because of its design and not due to any post training steps.

The main reason of preferring intrinsic interpretability methods to post hoc interpretability methods is that interpretability is neither an inevitable result of the discriminative power of a DNN, nor a prerequisite for it as demonstrated in [1]. Thus, it is required to ensure that the DNN learns the concepts during its training phase, instead of verifying if they have been learned post training. This is usually done by either inducing inductive biases during its training phase or by constraining the latent space of the DNN during its training phase.

This work aims to prove that Formal Concept Analysis (FCA) can be an effective method to discover the concepts that the intrinsically interpretable document classifier should learn so as to be able to explain its predictions. This is achieved by mapping the ambiguous idea of a concept to the mathematically defined notion of a formal concept, thereby allowing the creation of a conceptual hierarchy that is expressed via a concept lattice.

The intrinsically interpretable document classifier cannot directly utilize the hierarchical order information present in the concept lattices of document classes for its training, which necessitates the mapping of the formal concepts in a concept lattice to a graph topological space. The formal concepts of the concept lattices of each class of documents are treated as vertices of a directed, acyclic graph whose components are the concept lattices themselves. The vertices of the graph correspond to different formal concepts that belong to various classes, with the edges between vertices denoting the subconcept/superconcept relation. A binary classifier that is trained to detect the presence of edges among the formal concept vertices of document-class concept lattices, can be used to classify a test document by predicting if/where potential edges connect the test document vertex to the formal concept vertices found within the document-class concept lattices.

Finally, the formal concepts that have contributed the most toward the classification of a test document as belonging to a particular document class are determined by the number of edges that are predicted to connect between them and the test documents. The intents of such formal concepts reveal attributes that are more comprehensible to human beings.

The final aim of this work is to create a document classifier that is both highly interpretable and possesses relatively high classification performance. Such a model seeks to overcome the trade-off between model interpretability and performance, which asserts that predictive performance of an ML model is usually sacrificed for interpretability and vice versa; as stated in [2].

2. Related work

2.1. Concept-based explanations

The idea of concept-based explanations was first introduced in the seminal work of [3]. The work formally defines the problem of finding concepts as a interpretation function $g : E_m \rightarrow E_h$; that maps from a vector space E_m that defines the state of the DNN, to E_h a vector space in which human beings operate. The work introduces Concept Activation Vectors (CAVs) as a means of translation between E_m and E_h. A CAV is used to formally represent the notion of a concept in any layer of the DNN. A CAV is defined for a layer l of the DNN as a vector that is normal to the hyperplane separating the activations of a set of examples where the concept is present from a set of random examples. CAVs were used as a component of a method named Testing with CAVs (TCAV) which uses directional derivatives to measure the sensitivity of the predictions made by the DNN toward a concept that was learned by a CAV.

The original work while revolutionary for introducing the notion of concept-based explanations, still has some flaws. It could not inherently point toward important concepts, but could only respond to queries from the user about the significance of concepts, that must be supplied by the user themselves. The awareness and availability of well-defined concepts also affects the performance of the TCAV method, as deficiencies in either area lead to the possibility of there being a possibly infinite space of concepts from which to query. These flaws were addressed in the work of [4].

Since it is difficult to exactly define a concept, the work of [4] states three desirable properties that any concept-based explanation method must have to be comprehensible to human beings. Additionally, the work also provides the Automated Concept-based Explanation (ACE) method that can automatically identify concepts that are significant to the predictions made by a DNN. The particular example demonstrated in the work used a combination of image segmentation and image clustering to find concepts that are significant to the predictions made by a DNN trained to classify images.

Both of the previously mentioned concept-based explanation methods can be classified as post hoc interpretability methods. For reasons mentioned in Section 1, more recent concept-based explanation methods tend to belong to the class of intrinsic interpretability methods. One noteworthy example of such a concept-based explanation method is the work of [5]. This work introduced the notion of concept bottleneck models, that train on data points (x, c, y); where an input x is annotated with a human-specified concept c and a target y. Given x, the concept bottleneck model is trained to first predict the intermediate concept $\hat{c}$, which is then used to predict the target $\hat{y}$. A unique characteristic of concept bottleneck models is that they allow intervention on $\hat{c}$ by a domain expert, allowing them to edit $\hat{c}$ and propagate the corresponding changes to $\hat{y}$.

The work of [6] is also a prominent example of a concept-based explanation method that introduces a mechanism known as concept whitening. Concept whitening is implemented via a module inserted into a given layer of a DNN in order to constrain its latent space so as to represent target concepts, as well as extract them in a straightforward manner. Given a DNN classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ which has a hidden layer $\mathcal{Z}$, the classifier can be divided into the following two parts: a feature extractor $\Phi : \mathcal{X} \rightarrow \mathcal{Z}$ with parameter θ and a classifier $g : \mathcal{Z} \rightarrow \mathcal{Y}$

parameterized by ω. The goal of concept whitening is to learn Φ and g simultaneously such that

i the classifier $g(\Phi(.;\theta);\omega)$ accurately predicts the class.

ii the j^{th} dimension z_j of the latent representation **z** aligns with concept c_j

2.2. Interpretability via FCA

An example of using FCA for interpreting a DNN is the work of [7]. The main idea proposed was the generation of the architecture of a DNN based on the covering relation (the graph of the diagram) of a lattice obtained from either an antitone Galois connection (concept lattice) or a monotone Galois connection (giving rise to another type of a lattice), where every neuron can be interpreted as a concept. In the derived architecture of the DNN, the vertices of the DNN correspond to sets of similar objects with the similarity given by the set of their common attributes. The edges connecting the vertices also add to the interpretability of the DNN by denoting either concept generality (bottom-up) or conditional probability (top-bottom). This work utilizes ideas from the work of [7] in grouping similar objects as formal concept vertices based their common attributes.

3. Basic definitions

FCA is a branch of applied lattice theory that deals with deriving a concept hierarchy from a collection of objects and their attributes. The scope of the applicability of FCA to the coinciding attributes of the complete concepts extracted from ML models trained on tabular, text or sequential data can be best understood by first formally defining what a formal context, formal concept and a concept lattice are.

Definition 3.1: 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. An object g that is in a relation I with an attribute m is written as gIm.

Definition 3.2: For a set $A \subseteq G$ of objects, the **derivation operator** is defined as

$$A^{'} := \{m \in M \,|\, gIm \text{ for all } g \in A\}$$

(the set of attributes common to the objects in A). Correspondingly, for a set B of attributes, the **derivation operator** is defined as

$$B^{'} := \{g \in G \,|\, gIm \text{ for all } m \in B\}$$

Definition 3.3: A **formal concept** of the context (G, M, I) is a pair (A, B) with $A \subseteq G, B \subseteq M, A^{'} = B$ and $B^{'} = A$. A is called the **extent** and B is called the **intent** of the concept (A, B). $\mathfrak{B}(G, M, I)$ denotes the set of all concepts of the context (G, M, I).

Definition 3.4: If (A_1, B_1) and (A_2, B_2) are concepts of a context, (A_1, B_1) is called a **subconcept** of (A_2, B_2), provided that $A_1 \subseteq A_2$ (which is equivalent to $B_2 \subseteq B_1$). In this case (A_2, B_2) is the **superconcept** of (A_1, B_1) and it is possible to state that $(A_1, B_1) \leq (A_2, B_2)$.

The relation $\leq$ is called the **hierarchial order** (or simply **order**) of the concepts. The set of all concepts of (G, M, I) ordered in this way is denoted by $\underline{\mathfrak{B}}(G, M, I)$ and is called the **concept lattice** of the context (G, M, I).

4. Methodology

The methodology of this work consists of the following parts:

- Extracting the keywords from the training documents, to be used as attributes for building the training formal contexts for each document class.
- Building the concept lattice for each document class to be used for training the intrinsically interpretable document classifier.
- Validating the concept lattice built for each document class using a lazy FCA document classifier, which is similar to the one in [8] in order to classify documents via the concept lattice built for each document class.
- Mapping the training formal concepts of the document-class concept lattices to a graph topological space.
- Training a binary classifier to predict the potential edges between a test document vertex and the training formal concepts vertices in the document-class concept lattices, then an aggregate scoring function classifies the test document vertex according to the number of potential edges it has and to which training formal concept vertices the edges connect to.
- Revealing the training formal concepts along with their respective intents that contributed the most toward the test document vertex being predicted as belonging to a particular document class.

The relevant components of the methodology are illustrated via the toy example in Fig. 1.

4.1. Formal context creation

The classification performance of the intrinsically interpretable document classifier can only be reasonable if the formal contexts that are obtained for each class of the training data used to create the document-class concept lattices can adequately capture the underlying dependencies of the data, with minimum information loss. This can be validated by first testing the performance of a lazy FCA document classifier that uses the same document-class concept lattices that the intrinsically interpretable document classifier will use for training.

Each individual document is considered to be an object and a combined list of keywords extracted from all of the classes separately are considered to be its attributes. The keywords are extracted using the YAKE! algorithm [9], which was chosen because of its superior performance when compared to its contemporaries; as well as other conceptual scaling methods for text data. At test time, a test document is transformed into a formal context object with its attributes being the combined list of keywords extracted from all of the classes separately for all of the training documents.

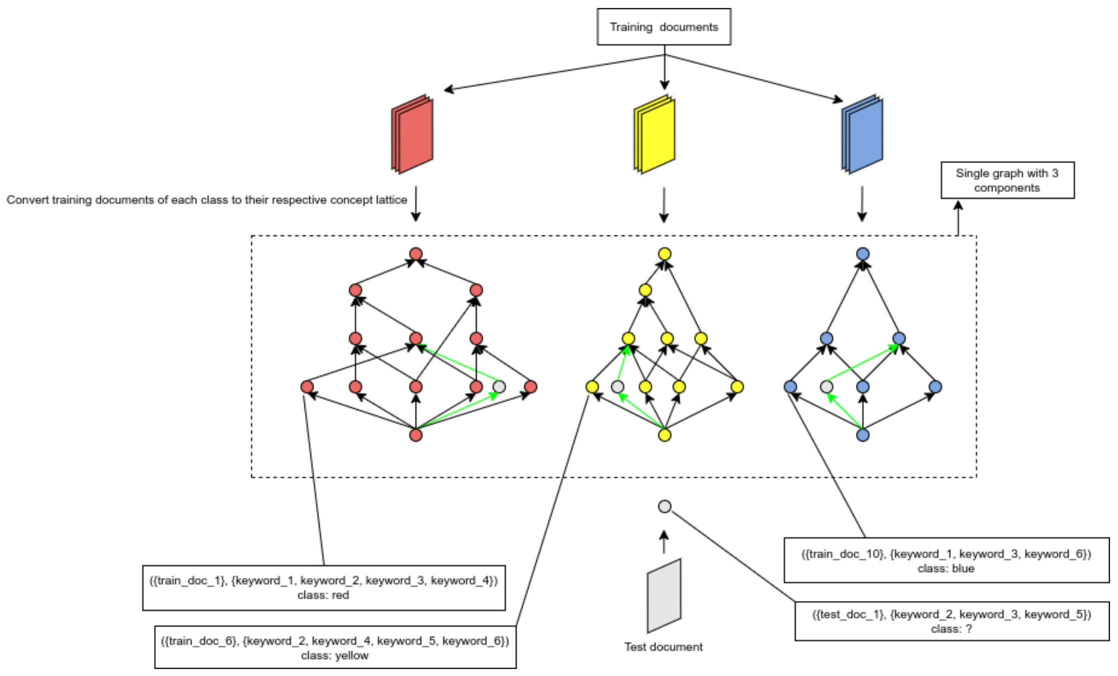
Training documents
Convert training documents of each class to their respective concept lattice
Single graph with 3 components
({train_doc_1}, {keyword_1, keyword_2, keyword_3, keyword_4})
class: red
({train_doc_6}, {keyword_2, keyword_4, keyword_5, keyword_6})
class: yellow
Test document
({train_doc_10}, {keyword_1, keyword_3, keyword_6})
class: blue
({test_doc_1}, {keyword_2, keyword_3, keyword_5})
class: ?

Figure 1: First, documents of various classes (depicted in red, yellow and blue) are grouped together to create their respective concept lattices for training. The concept lattices are treated as components of a single, acyclic, directed, training graph whose vertices represent formal concepts. A binary classifier is first trained to predict the presence of edges (depicted in black) among the formal concept vertices of the training graph and then is used to predict if/where any potential edges (depicted in green) connect a test document vertex (depicted in grey) to the formal concept vertices of the training graph. An aggregate scoring function assigns the test document vertex to a class according to the formal concepts and the frequency of the potential edges between the test document vertex and the formal concept vertices of that particular document-class concept lattice.

4.2. Lazy FCA document classifier

The aim of the lazy FCA document classifier is to classify test documents by calculating a set of classification scores $\{S_{L_i}\}$ for each test document. Each document-class concept lattice L_i is built using the formal context obtained from the training documents of the i^{th} document class. A test document is classified as belonging to the class that has the highest classification score S_{L_i}. The classification scores of a lazy FCA document classifier can be defined in many different ways, this particular work uses the following classification score:

$$S_{L_i} = \frac{1}{|t(L_i)|} \sum_{j \in t(L_i)} |int_j \sqcap int_{test}| \times |\text{ext}(j)| \Big[|int_j \sqcap int_{test}| \geq 0.5 \times |int_{test}| \Big] \tag{1}$$

where int and ext stand for intent and extent, respectively. The S_{L_i} score first checks if the number of attributes present in the intersection of the attributes for an intent of a formal concept j and the attributes for an intent of the test document formal context object exceed a threshold that is greater than or equal to half of the total number of attributes present in the intent of j. If the aforementioned condition (written in the Iverson bracket of equation (1)) is satisfied, the number of attributes in the intersection that satisfy the condition is weighted by the extent of the formal concept j, which is used here as its interestingness measure [10].

This operation is done and the sum incremented for every formal concept j that belongs to the set $t(L_i)$. The threshold function t that is applied to a document-class concept lattice L_i is defined as $t(L_i) = \{j \in L_i \mid f(j) \leq \tau\}$ (τ being a hyperparameter). The function f is defined for a formal concept j as $f(j) = |\{ext_k \mid int_j \in int_k\}|$; with training documents ({ext_k}, {int_k}), called “counterexamples”, belonging to any class other than the one that the formal concept j belongs to. The function f therefore, finds the number of counterexamples that a formal concept j has.

4.3. Intrinsically interpretable document classifier

The functioning of the intrinsically interpretable document classifier is explained with reference to the toy example illustrated in Fig. 1. Each of the documents belong to a class i, where $i \in$ {red, yellow, blue}. For the purpose of training the intrinsically interpretable document classifier, similar to the process in Section 4.2; three document-class concept lattices L_i are built for each class i using their respective training documents.

Referring to Fig. 1, there are three document-class concept lattices: L_{red}, L_{blue} and L_{yellow}. Each formal concept c_{ij} that belongs to the i^{th} document-class concept lattice L_i, has its own extent and intent pair ({ext_{ij}}, {int_{ij}}), where ext_{ij} is a set of documents and int_{ij} is a set of keywords that are present in all of the documents that constitute its respective extent ext_{ij}, e.g., ({train_doc_1, train_doc_2, train_doc_3}, {keyword_1, keyword_2}).

With reference to Fig. 1, the three document-class concept lattices corresponding to the three classes can be considered as three components of a single, acyclic, directed, training graph $G_{train} = (V_{train}, E_{train})$. Each training vertex v_{ij} of G_{train} is a formal concept c_{ij} = ({ext_{ij}}, {int_{ij}}) that belongs to class i. A training edge $e \in E_{train}$ (depicted in black) that connects the vertices v_{ij} to each other is considered equivalent to an (implied) arc that connects two elements in a lattice diagram. Thus, the training graph G_{train} maintains the order relation, i.e., the superconcept/subconcept relation, among its training vertices v_{ij} (which represent the formal concepts c_{ij}).

A test document (depicted in grey) belonging to an unknown class i_{test}, is first converted to a test vertex v_{test} with its own extent and intent pair ({ext_{test}}, {int_{test}}) by means of the same formal context creation process described in Section 4.1. It is important to note that new keywords are not generated from the test documents so as to be used as the attributes of int_{test}, rather the keywords obtained from the training documents during the training phase are used as the attributes of int_{test}. Consequently, ext_{test} of a singular test document v_{test} is just a singleton, consisting of its own object id (file name). Thus, the extent and intent pair of a test document v_{test} is of the form ({test_doc_id}, {keyword_1, keyword_2, keyword_3, keyword_4,...}).

Using the topology of the training graph G_{train}; as well as the concepts related to its vertices v_{ij}, it is possible to predict all of the potential edges e_{test} (depicted in green) between the test vertex v_{test} and the training vertices v_{ij}. This is done in order to infer the class i_{test} of the test vertex v_{test}, based on the particular formal concepts c_{ij} (represented by the training vertices v_{ij}) that the potential edges e_{test} of v_{test} may connect to.

A DNN is trained as a binary classifier on the concatenated intents of pairs of training vertices v_{ij} in order to predict whether an edge e either exists between them or not. Then for predicting if/where the possible edges e_{test} of a test vertex v_{test} may connect to, the DNN takes as input concatenated intents of all pairs of training vertices v_{ij} and the test vertex v_{test}.

The specific architecture of the DNN used in this work is comprised of 6 fully connected layers with 192, 128, 64, 32, 16 and 1 neuron(s). Batch normalization is used after the second and fourth layers during the training phase. Dropout is used after the third and fifth layers during the training phase with respective probabilities of 0.3 and 0.2. The ReLU activation function is used after every layer. The DNN has a weight decay of 5e-4, uses a binary cross-entropy loss function and is optimized via the Adam optimization algorithm [11]; having a learning rate of 0.001 and a learning rate decay multiplicative factor of 0.1.

The aggregation function computes a score A_{L_i} for every document-class concept lattice L_i and infers the class i_{test} of the test vertex v_{test} as belonging to the document class whose concept lattice has the highest aggregate score A_{L_i}.

$$A_{L_i} = \frac{1}{|L_i|} \sum_{j \in L_i} E(v_{test}, v_{ij}) \times p(v_{ij}) \tag{2}$$

where $E(v_t, v_{ij}) = \begin{cases} 0, & \text{if no edge exists between } v_{test} \text{ and } v_{ij} \\ 1, & \text{if an edge exists between } v_{test} \text{ and } v_{ij} \end{cases}$

and $p(v_{ij}) = \frac{1}{f(v_{ij})+1}$

The function $f(v_{ij})$ counts the number of counterexamples of the formal concept v_{ij}, was previously defined in Section 4.2 and is a part of the penalty function p.

The rationale for creating such an aggregate score A_{L_i} for each document-class concept lattice, in which the function $E(v_{test}, v_{ij})$ is an essential component can be stated as follows:

1. It is not possible to accurately predict the exact training vertex v_{ij} that the predicted edge should connect to, in order to maintain the hierarchical order of formal concepts within the corresponding document-class concept lattice. This is particularly important for reasons of generating explanations as further elaborated in point 3.

2. Since a relatively simple method of representing attributes of the intents is used in this work, i.e., presence or absence of keywords, many pairs of the training vertices v_{ij} and the test vertex v_{test} can have the exact same set of intents. Thus, it is beneficial to maximize the number of edge prediction tasks for both the training (as a form of auxiliary training data) and the testing of the intrinsically interpretable document classifier (in order to bolster the aggregate scoring function A_{L_i}).

3. It is necessary to find those training vertices v_{ij} that may connect to that text vertex v_{test} in order to quantify the contribution of a formal concept v_{ij} toward classifying v_{test} as belonging to a particular class. This is illustrated by the use of the function $E(v_{test}, v_j)$ in computing the metric $Q_i(v_j)$.

4.4. Quantifying the contribution of a formal concept

The contribution of a formal concept v_j toward classifying test documents v_t as belonging to a particular document class i can quantified by measuring the contribution of any predicted edges between them toward the aggregate score A_{L_i} and is defined as $Q_i(v_j) = \sum_{t \in \text{test}_i} E(v_t, v_j) \times p(v_j)$.

4.5. Dataset

The chosen dataset [12] for use in this work is a subset of a larger dataset consisting of newsgroup documents [13]. There are a total of 1000 documents, which are divided into 10 classes; with each class having 100 documents. The classes are disjoint, which means that they are no documents that belong to more than one class. After the formal context creation step specified in Section 4.1, each document has a set of attributes that denote the presence or absence of 96 unique keywords.

5. Experiment and results

The relatively small size of the dataset having only 1000 documents necessitates the need of using more data for training as compared to testing. Thus all experiments will be conducted with a 90:10 training to test split ratio. There will be 90 training documents for each class and 10 test documents for each class. The table below describes some characteristics of the training set after creating the document-class concept lattices.

Table 1
Table describing some characteristics of the document-class concept lattices obtained from the training documents of each class.

Class	No. of concepts	No. of attr.	Avg. no. of attr. per concept
Business	551	96	3.851
Entertainment	337	96	3.689
Food	185	96	3.537
Graphics	206	96	3.893
Historical	984	96	4.996
Medical	211	96	5.806
Politics	607	96	3.955
Space	552	96	4.842
Sport	297	96	4.513
Technology	986	96	4.062

5.1. Lazy FCA document classifier performance

As demonstrated by the table below, the lazy FCA document classifier demonstrates reasonable classification performance, across nearly all classes; with only certain classes showing poor performance. This means that the document-class concepts lattices were able to reasonably capture the underlying data dependencies in each class and map it to the conceptual hierarchy

Table 2
Table describing the classification performance of the lazy FCA document classifier described in Section 4.2

Class	Precision	Recall	F1-Score	Support
Business	1.00	0.30	0.46	10
Entertainment	1.00	0.50	0.67	10
Food	0.91	1.00	0.95	10
Graphics	0.86	0.60	0.71	10
Historical	0.35	0.60	0.44	10
Medical	0.26	0.50	0.34	10
Politics	0.89	0.80	0.84	10
Space	0.86	0.60	0.71	10
Sport	0.64	0.90	0.75	10
Technology	0.75	0.60	0.67	10
Accuracy			0.64	100
Macro avg	0.75	0.64	0.65	100
Weighted avg	0.75	0.64	0.65	100

found in the concept lattices of each class. It is important to note that without this happening, there would be no reason to believe that the intrinsically interpretable document classifier would have good classification performance. The lazy FCA document classifier was run using hyperparameter value $\tau = 20$.

5.2. Intrinsically interpretable document classifier performance

Table 3
Table describing the classification performance of the intrinsically interpretable document classifier described in Section 4.3.

Class	Precision	Recall	F1-Score	Support
Business	0.86	0.60	0.71	10
Entertainment	0.78	0.70	0.74	10
Food	0.64	0.90	0.75	10
Graphics	1.00	0.60	0.75	10
Historical	0.71	0.50	0.59	10
Medical	0.67	0.60	0.63	10
Politics	0.83	1.00	0.91	10
Space	0.60	0.90	0.72	10
Sport	0.88	0.70	0.78	10
Technology	0.69	0.90	0.78	10
Accuracy			0.74	100
Macro avg	0.77	0.74	0.74	100
Weighted avg	0.77	0.74	0.74	100

As demonstrated by the table above, the intrinsically interpretable document classifier demonstrates a marked improvement in classification performance, across all classes; as compared to the lazy FCA document classifier.

5.3. Maximally contributing formal concepts

Table 4
Table describing the formal concepts that contributed the most toward classifying test documents as belonging to a particular class as described in Section 4.4

Class	Intent	$Q_i(v_j)$
Business	('asia', 'firm', 'sale')	3
Entertainment	('film', 'fly')	5
Food	('cup', 'inch', 'vegetable')	9
Graphics	('psp',)	5
Historical	('citizenship', 'french', 'war', 'world')	6
Medical	('medical', 'wvnvms')	5
Politics	('bbc', 'government', 'local')	8
Space	('earth', 'put', 'space')	6
Sport	('big', 'london', 'race', 'thing', 'world', 'year', 'york')	5
Technology	('firm', 'junk', 'virus')	5

.

The table above lists the formal concepts v_j that have the maximum value for $Q_i(v_j)$ for a document class i. The attributes of the intents of such formal concepts are intuitively logical and appear in at least half of all but one of the test documents of each class.

6. Conclusion and future work

An intrinsically interpretable document classifier with moderately high classification performance that uses properties of the graph of the covering relation of a concept lattice (lattice diagram) was proposed. The intrinsically interpretable document classifier is able to quantify which formal concepts contributed the most toward the classification of a test document. Future work can focus on using more advanced methods for representing complex data such as pattern structures [14, 15] as well as using more sophisticated methods to take into account the assortativity of the graph vertices.

Acknowledgments

The work of Sergei O. Kuznetsov was supported by the Russian Science Foundation under grant 22-11-00323 and performed at HSE University, Moscow, Russia.

References

[1] D. Bau, B. Zhou, A. Khosla, A. Oliva, A. Torralba, Network dissection: Quantifying interpretability of deep visual representations, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, IEEE Computer Society, 2017, pp. 3319–3327. URL: https://doi.org/10.1109/CVPR.2017.354. doi:10.1109/CVPR.2017.354.

[2] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, Information Fusion 58 (2020) 82–115. URL: https://www.sciencedirect.com/science/article/pii/S1566253519308103. doi:https://doi.org/10.1016/j.inffus.2019.12.012.

[3] B. Kim, M. Wattenberg, J. Gilmer, C. J. Cai, J. Wexler, F. B. Viégas, R. Sayres, Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV), in: J. G. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of *Proceedings of Machine Learning Research*, PMLR, 2018, pp. 2673–2682. URL: http://proceedings.mlr.press/v80/kim18d.html.

[4] A. Ghorbani, J. Wexler, J. Y. Zou, B. Kim, Towards automatic concept-based explanations, in: H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 9273–9282. URL: https://proceedings.neurips.cc/paper/2019/hash/77d2afcb31f6493e350fca61764efb9a-Abstract.html.

[5] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, P. Liang, Concept bottleneck models, in: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 5338–5348. URL: http://proceedings.mlr.press/v119/koh20a.html.

[6] Z. Chen, Y. Bei, C. Rudin, Concept whitening for interpretable image recognition, Nat. Mach. Intell. 2 (2020) 772–782. URL: https://doi.org/10.1038/s42256-020-00265-z. doi:10.1038/s42256-020-00265-z.

[7] S. O. Kuznetsov, N. Makhazhanov, M. Ushakov, On neural network architecture based on concept lattices, in: M. Kryszkiewicz, A. Appice, D. Slezak, H. Rybinski, A. Skowron, Z. W. Ras (Eds.), Foundations of Intelligent Systems - 23rd International Symposium, ISMIS 2017, Warsaw, Poland, June 26-29, 2017, Proceedings, volume 10352 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 653–663. URL: https://doi.org/10.1007/978-3-319-60438-1_64. doi:10.1007/978-3-319-60438-1_64.

[8] S. O. Kuznetsov, Fitting pattern structures to knowledge discovery in big data, in: P. Cellier, F. Distel, B. Ganter (Eds.), Formal Concept Analysis, 11th International Conference, ICFCA 2013, Dresden, Germany, May 21-24, 2013. Proceedings, volume 7880 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 254–266. URL: https://doi.org/10.1007/978-3-642-38317-5_17. doi:10.1007/978-3-642-38317-5_17.

[9] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, A. Jatowt, Yake! keyword extraction from single documents using multiple local features, Inf. Sci. 509 (2020) 257–289. URL: https://doi.org/10.1016/j.ins.2019.09.013. doi:10.1016/j.ins.2019.09.013.

[10] S. O. Kuznetsov, T. P. Makhalova, Concept interestingness measures: a comparative study, in: S. B. Yahia, J. Konecny (Eds.), Proceedings of the Twelfth International Conference on Concept Lattices and Their Applications, Clermont-Ferrand, France, October 13-16, 2015, volume 1466 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015, pp. 59–72. URL: http://ceur-ws.org/Vol-1466/paper05.pdf.

[11] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL: http://arxiv.org/abs/1412.6980.

[12] J. Baxter, (10)dataset text document classification, 2020. URL: https://www.kaggle.com/datasets/jensenbaxter/10dataset-text-document-classification.

[13] T. Mitchell, 20 newsgroups, 1999. URL: http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html.

[14] B. Ganter, S. O. Kuznetsov, Pattern structures and their projections, in: H. S. Delugach, G. Stumme (Eds.), Conceptual Structures: Broadening the Base, 9th International Conference on Conceptual Structures, ICCS 2001, Stanford, CA, USA, July 30-August 3, 2001, Proceedings, volume 2120 of *Lecture Notes in Computer Science*, Springer, 2001, pp. 129–142. URL: https://doi.org/10.1007/3-540-44583-8_10. doi:10.1007/3-540-44583-8_10.

[15] S. O. Kuznetsov, Scalable knowledge discovery in complex data with pattern structures, in: P. Maji, A. Ghosh, M. N. Murty, K. Ghosh, S. K. Pal (Eds.), Pattern Recognition and Machine Intelligence - 5th International Conference, PReMI 2013, Kolkata, India, December 10-14, 2013. Proceedings, volume 8251 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 30–39. URL: https://doi.org/10.1007/978-3-642-45062-4_3. doi:10.1007/978-3-642-45062-4_3.