Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2022) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

26th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2022)

Combining an explainable model based on ontologies with an explanation interface to classify images

Matthieu Bellucci*, Nicolas Delestre, Nicolas Malandain, Cecilia Zanni-Merk

Normandie Université, INSA Rouen, LITIS, Rouen 76000, France

Abstract

Numerous explainability methods have appeared thanks to the surge of popularity of the explainable AI (XAI) domain. The DARPA depicted an explainable system, where an explainable model interacts with an explanation interface to generate explanations adapted to a user. We propose an explainable image classification system that follows this combination of explainable model and explanation interface as described by the DARPA. It takes advantage of the explainability of ontologies as well as the performance of machine learning models. This system is able to predict the class and properties of an object in an image. The results of this classification system are displayed in an explanation interface to help users understand and analyze the predictions of the proposed system. Our system exploits an ontology to build models that classify an object and its properties. The class and properties predicted by these models are instantiated in the ontology and added as assertions to an individual in order to verify the consistency of these predictions. Therefore, the system is able to warn the user when a prediction is uncertain and explain why, by using the ontology. These capacities will help users trust this system and better understand the predictions.

Keywords: XAI; Ontology; Explanation

1. Introduction

Artificial Intelligence (AI) is being adopted in many critical domains such as healthcare or justice. Decisions coming from statistical models have a significant impact on our lives. That is why there is a real need to explain how and why these decisions were made. That is the goal of explainable artificial intelligence (XAI). These explanations need to be tailored to the level and domain of expertise of the explainee. Many explainability methods have already been developed to explain black-boxes [1] or create interpretable models [2, 3]. Gunning et al. [4] provided an architecture for an explainable system composed of two main elements: an explainable model and an explanation interface. These two elements interact with each other to provide accurate predictions and generate explanations.

* Corresponding author.
E-mail address: matthieu.bellucci@insa-rouen.fr

In this paper, we propose an explainable system that follows the design described by Gunning et al [4]. It uses knowledge in the form of an ontology, to train, make predictions and explain these predictions. Several image classifiers are tasked with recognizing some properties in an image. These properties are linked to some classes in ontology, which allows the system to classify an object in an image based on what the models observed. A logical reasoner is then used to check the consistency of this prediction, based on the class prediction and the observed properties. This consistency check enables the system to tell the user when and why a prediction is uncertain. We designed an explanation interface that presents the results of this explainable model so that the general public can understand its predictions.

In section 2 we review the literature, especially the use of semantic web technologies to build explainable systems and guidelines for designing an explainable user interface. Then in section 3, we describe our explainable model and explanation interface. In section 4 we apply and test this system to the case of musical instruments classification. Finally in section 5, we discuss this approach, its limitations and future works.

2. Related works

There exists a large variety of explainability techniques, that have been categorized and compiled into toolkits [1, 5]. These toolkits or methods take the form of Python packages that are not accessible to the general public. Moreover, most of them are post hoc techniques that aim at explaining black boxes. Post hoc techniques are explainability techniques that use auxiliary methods to explain a model after it is trained, according to [1]. Rudin [6] advocates against using these techniques, arguing that their explanations are not reliable and can be misleading. In the same paper, it is discussed that interpretable models should be preferred.

To our knowledge, few explanation interfaces that generate explanations adapted to the general public exist. Furthermore, few existing interfaces are freely accessible to the public. Among open-source interfaces, the What-If Tool developed by Wexler et al. [7] is very useful for AI experts to explore and diagnose their data and models but it is not easily accessible to anyone else. Nevertheless, many scholars have studied how to develop accessible and user-friendly explanation interfaces. Sokol et al. [8] designed fact sheets to assess explainability methods. They discuss five dimensions: functional and operational requirements, usability, security and validation of the methods. Their goal is to provide a way to evaluate and categorize explainability techniques. Likewise, Amershi et al. [9] present 18 guidelines for human-AI interaction. Chromik et al. [10] review the principles for explanation user interfaces (XUI). Their work is guided by two research questions: how to characterize different concepts of interaction in XAI and what design principles for interactive XUIs can be observed. They describe different kinds of interaction between a human and an AI. Interaction as embodied action is of interest to us because it is about cooperation between humans and computers to make decisions, which is what we want to achieve. They also introduce the concept of exploratory XUI, where the user can explore the interface to get different explanations on different levels, i.e. global explanations to understand how the system works or local explanations to understand the system's prediction. Finally, Liao et al. [11] surveyed the current state of the explainability in various AI products, identified user needs and compiled an XAI question bank which will help design user-centred XAI applications.

As we mentioned earlier, Rudin [6] says that interpretable models should be preferred. Lipton [12] addresses the issue of defining what interpretability is. Bellucci et al. [13] discuss the larger issue of the XAI terminology and the lack of consensual explicit definitions. They propose definitions for different XAI terms that we use in this paper. They define interpretability as "the ability of a system to be seen, understood and studied by a user with a reasonable cognitive effort". Ontologies and specifically semantic web technologies represent knowledge in both a human-understandable and machine-readable way [14].

Using such technologies may help design a system that can be seen, understood and studied by a human, making it interpretable. Seeliger et al. [15] provide a literature review of semantic web technologies for explainable machine learning models. They argue that explainability is dependent on the use of domain knowledge and that semantic web technologies might be a key to achieving truly explainable AI systems.

Semantic web technologies can also be used to generate explanations. Futia and Vetrò [16] examine the advantages of integrating knowledge graphs into deep learning models. They claim that combining them enables the creation of interactive and cross-disciplinary explanations. Halliwel et al. [17] propose a method to find the most intuitive explanations using facts of an ontology and evaluate the quality of these explanations. Rožanec et al. [18] create feature-based explanations to extract the most important features and then use an ontology to get their context and retrieve higher level concepts that describe them. In this paper, the context corresponds to pieces of data and events that may influence the prediction. The context and higher-level concepts are then used to create an explanation.

Gunning et al. [4] provide a framework to design explainable systems, as we have already discussed. However, they do not mention the use of semantics to design these explainable models or explanation interfaces. We also identified a lack of interfaces that use the different explanation toolkits already developed and make them accessible to the general public. To address these issues, we propose an explainable system that incorporates semantic web technologies, in the form of a web semantic ontology, to advance towards an interpretable model. Then, we describe a modular design for an explanation interface that presents our system's predictions and that will be able to generate explanations using different techniques and adapt these explanations to the user.

3. Ontology-based Image Classifier

The paper presents an ontology-based image classifier, which combines machine learning with an ontology. As was mentioned in section 1, this classifier uses the architecture described in [4], that is to say, an explainable model interacting with an explanation interface. The goals of this architecture are to make an explainable system for image classification that is trustworthy and robust. Specifically, the system is able to detect when a prediction is uncertain and inform the user if the prediction can be trusted and why. The design of an explanation interface that presents the results of the explainable model are then discussed. The main goal of this interface is to make it accessible to the general public while still providing a sufficient amount of information.

The system is composed of two main components: an ontology with a logical reasoner and a set of machine learning models, each capable of detecting one property visible in the image. For instance, one model will be tasked with detecting the texture in the image and another with detecting particular shapes. Then, a machine learning model that we call the *global* classifier is trained to directly classify the data. To make an inference, an image is given to every model. An individual with the class and properties detected by these models is added to the ABox of the ontology. Finally, a logical reasoner is used to verify the consistency of the ontology. If the ontology is consistent, the prediction is considered valid. Otherwise, the user is warned via the explanation interface that the prediction is uncertain. The explanation interface shows the result of the classification, the consistency of the prediction and gives the option to observe the results of each model. In case of an inconsistency, the interface gives the reasons for this inconsistency.

In this section, we first define the notations used in the rest of the paper to describe ontology elements. Then, we discuss the ontology architecture needed to make this system work. Afterwards, we present the inference processes. Finally, we describe the explanation interface that presents the results of a prediction from our system.

3.1. Notation

In the following, ontology related terms such as domain and range are defined in the W3C recommendation for the Web Ontology Language (OWL) [19]. The classes and object properties of an ontology are written in the following style: `MyClass` for classes and *`myObjectProperty`* for object properties. Class restrictions are presented as such: "*`myObjectProperty`* constraint `TargetClass`", where `TargetClass` is part of the range of *`myObject-Property`*. In this restriction, we refer to *`myObjectProperty`* as the property of the restriction and `TargetClass` as its target or target class. Negative restrictions are restrictions that mention what a class is not. They take the form "Not *`myObjectProperty`* constraint `TargetClass`". Model refers to any machine learning model, such as a neural network or a decision tree. Concerning XAI terms, we use the definitions proposed in [13].

3.2. Ontology building

This system uses the open-world assumption and the Web Ontology Language version 2 [19] (OWL 2). Since we are working with images, the visible properties are highly dependent on the angle of the picture. We cannot assume that a property is missing when it is not visible. That is why we apply the open-world assumption; it does not consider that missing properties are proof of their absence.

In order to exploit the dataset, object properties that describe observable characteristics in the data should be added to the TBox of the ontology. These observable characteristics or properties are declared as subproperties of *observableProperty* in the ontology, so that the system can automatically identify and exploit them. The domain of these properties is the classes present in the dataset. The range can be any class or set of classes defined in the ontology.

The classes must be defined with restrictions whose properties are subproperties of *observableProperty*. The class definitions should be as precise as possible to maximize the accuracy and robustness of the system, as well as provide clearer explanations when an inconsistency is detected. Indeed, the more restricted a class is, the more likely an error in the prediction will lead to an inconsistency. Negative restrictions should also be used whenever possible to help the system find inconsistencies.

To summarize, class definitions should use subproperties of *observableProperty* in restrictions and be as precise as possible, using negative restrictions when feasible. Algorithm 1 illustrates a class definition for wooden chairs where its texture is an observable property. Here, wooden chairs are made of wood and cannot be made of metal.

Algorithm 1 Example of class definition, using OWL2 Manchester Syntax [20]

```
ObjectProperty: hasTexture, SubPropertyOf: observableProperty
Class: WoodenChair, SubClassOf: Chair that hasTexture some Wood,
SubClassOf: not hasTexture Metal
```

In order to make the explanations more understandable, the ontology should contain as much information as possible about the different classes and observable properties. Multiple descriptions and labels should be given as `owl:Annotation` for the same element of the TBox, corresponding to different levels and domains of expertise. Therefore, the explanation interface can exploit these descriptions and labels to generate explanations that are adapted to the user.

Some textual patterns may also be given to describe restrictions with the same object property. For instance, let us observe the restrictions with the property *hasTexture*, such as "*hasTexture* `Metal`". If this restriction is written in this format, some users may be discouraged to understand its meaning. A textual pattern in the definition of the property *hasTexture* will enable the interface to create a more user-friendly representation of this restriction. Such textual patterns should remain simple and short, such as "The object in the image has a `Metal` texture".

3.3. Inference

To make a prediction, the system uses one machine learning model per *observableProperty*. For a given property, a model takes an image as input and outputs a vector that associates a probability with each class in the range of this property. This probability corresponds to the chances of a class to be the target of a restriction with the studied property. The class is predicted by another machine learning model called *global* classifier. Algorithm 2 describes the different steps of an inference. Functions with an uppercase first letter are functions that modify the ontology, as described in functional syntax [21]. The other functions are designed by us.

The inference process computes the predictions of each model, adds the corresponding restrictions to a new individual in the ABox of the ontology and checks the consistency with this new individual. The class of the individual is given by the *global* classifier. If a prediction is inconsistent, the explanation interface will clearly show it, thus warning the user when a prediction is uncertain, which is a requirement of explainable systems according to Gunning et al. [4]. We believe that the ability of the system to verify the consistency of its own predictions makes it easier for a user to trust it. It is difficult to assess this gain in trust because of the subjectiveness of trust. However we argue that this ability is a step towards a trustworthy system. Moreover, the use of multiple independent statistical models and the consistency check to classify a single image makes the system more robust than using a only one classifier. To increase this robustness, we recommend using different models for each properties. Two parameters, $threshold^+$ and $threshold^-$ are introduced. They take advantage of the open-world assumption. Indeed, if a probability is in-between these thresholds, the system does not add a restriction and considers that the information is missing.

As discussed in section 2 this architecture allows the generation of different types of explanations thanks to its use of an ontology. The explanation interface will extract all relevant information from this explainable model to present and explain its results. This information include the main prediction, the results of every property model and the consistency of the ontology after adding the new individual in its ABox. It also allows the interface to generate

Algorithm 2 Inference algorithm

```
function INFER($x, onto, observablePropertiesSet, threshold^+, threshold^-$)    ▷ $x$ a data point, $onto$ the ontology
Ensure: $isConsistent(onto)$
    $class \leftarrow globalClassifier(x)$    ▷ Corresponds to the class predicted by the global classifier
    $Declaration\ (NamedIndividual(indiv))$
    $ClassAssertion\ (class, indiv)$
    for all $property$ in $observablePropertiesSet$ do
        $y \leftarrow propertyClassifier(property, x)$    ▷ $y$ is the vector of predictions.
        for $i \leftarrow 0, length(y)$ do
            $Declaration\ (NamedIndividual(target_i))$
            $ClassAssertion\ (class_i, target_i)$    ▷ $class_i$ is the $i$-th class in the labels for the classifier of $property$.
            if $y_i \geq threshold^+$ then
                $ObjectPropertyAssertion(property, indiv, target_i)$
            else if $y_i \leq threshold^-$ then
                $ClassAssertion\ (Not(property, class_i), indiv)$
            end if
            if not $isConsistent(onto)$ then
                return $False$
            end if
        end for
    end for
    return $True$
end function
```

truthful logic explanations using a logical reasoner. Finally, any machine learning model can be used with this system. When interpretable models are applied, the entire system becomes fully interpretable as is recommended by Rudin [6].

3.4. Explanation interface

We designed an explanation interface that displays the results of the system. This is the first part of a broader explanation interface capable of presenting different explanations and adapting to different users. The goal of the interface is to communicate with the explainable model to gather relevant information and interact with the user to provide tailored explanations. It corresponds to the concept of exploratory XUI as discussed in section 2. Chromik et al. [10] describe an XUI as a tool to present and adapt *raw explanations* to a user. These *raw explanations* come from explainability techniques that do not seek to adapt to different levels of expertise. Another objective of the interface is therefore to get these *raw explanations* and translate them into understandable explanations for the current user. The interface is designed to be modular so that we can leverage the power of other interfaces such as the What-If Tool [7].

We designed this interface to be used as a decision-making assistant where the final decision is made by the user. That is why our interface displays the key information used by the system: the outputs of the different models, the output from the *global* classifier, the different thresholds and the consistency of the prediction.

The prediction from the *global* classifier and its consistency are in a written form. When the prediction is detected as inconsistent, the problematic restrictions are shown to the user as a list. The user can further analyze the results by observing the output of each model. These outputs are displayed as colored bar charts. The color is chosen depending on whether a restriction is added. When the output is greater than $threshold^+$, the bar is colored in green to indicate that a restriction was added. When the output is lower than $threshold^-$, the bar is colored in red to indicate that a negative restriction was added. Finally, when the output is between these two thresholds, the bar is colored in orange, indicating that no restriction was added. The name of the target classes is also colored based on the consistency of the restriction.

As of now, the interface is static but already contains exploratory features in the form of the different charts for each property. Results are presented as a mixture of visual and textual explanations. The explanations are truthful with respect to the predictive model because they directly use the system's output to generate the explanations.

This interface is the first step towards a fully-fledged explanation interface, that will incorporate different types of explanations, such as counterfactual explanations or logic-based explanations. It will provide different explanations based on the user's expertise. Recommendations from [8, 9, 10] will be implemented in order to propose an interactive exploratory interface with multiple classes of explanations. We plan to distinguish three levels of expertise: AI expert, domain expert and non-expert i.e. someone who is not an AI or domain expert. The interface can therefore be used for different purposes:

- Diagnose the system for an AI expert.
- Assist in the decision making process for a domain expert.
- Explain the decision process to a non-expert who is affected by the decision.

We show an image of this interface and describe it further in section 4, figure 1. The interface is implemented as a website, using the *Model-View-Controller* design pattern, where the *Model* handles the predictions and computes the *raw explanations*. The *View* is what the user sees, it converts information received from the *Model* into explanations adapted to the user. Finally, the *Controller* allows the user to interact with the interface and helps the *View* to adapt the explanations to the user's level and domain of expertise. We chose this design pattern because it conveniently separates the different tasks of an explainable system. Thus, the *Model* can be used independently from the rest if no explanation is necessary. Likewise, the *View* and *Controller* may be used with different explainable models. We also decided to use a web format so that the interface is accessible and intuitive to everyone, on many devices, such as a computer or a mobile phone.

4. Experimentation

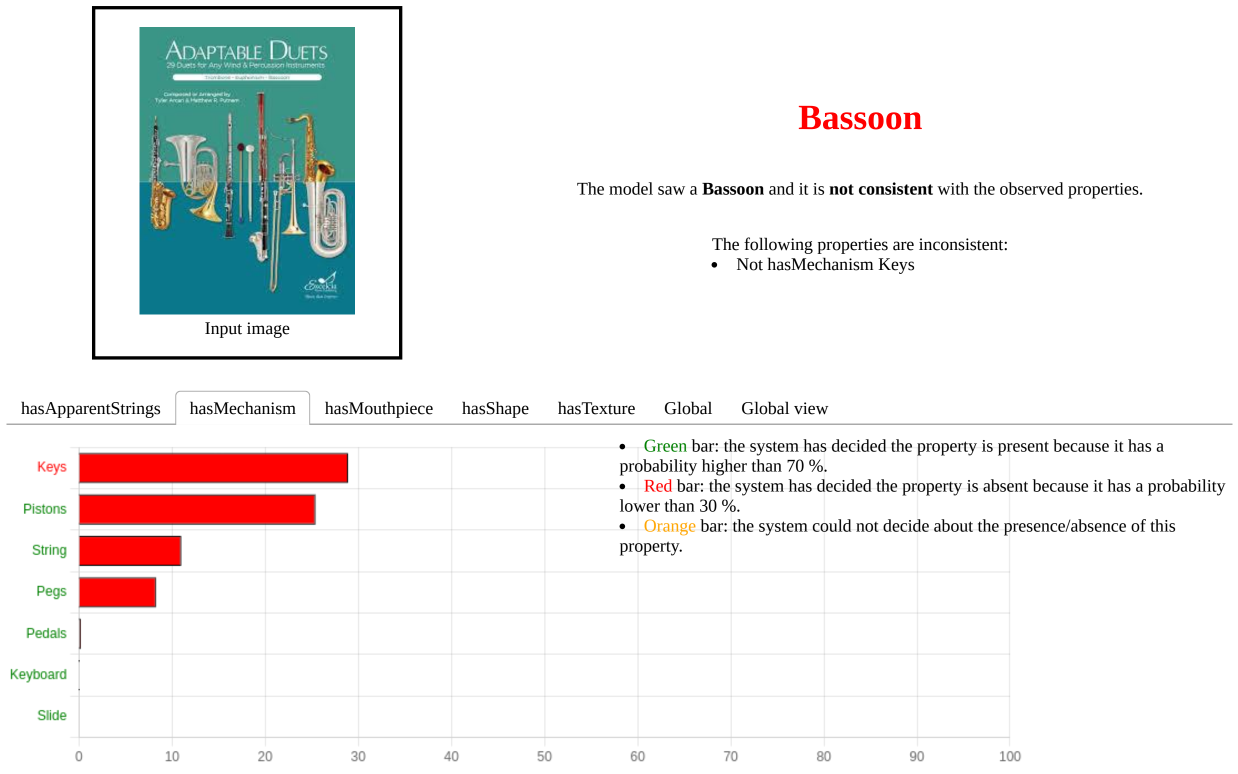
We evaluate this system on a musical instrument classification task. The task is to determine the type of musical instrument in an image. We have designed an ontology based on a simple hierarchy of instrument families. We added 5 subproperties of $\mathit{observableProperty}$ to its TBox. These observable properties are based on what is visible in the majority of the images, such as the texture of the instruments or their shape. The images were gathered from Google Images and were manually checked. There is a total of 20 classes, with approximately 250 images per class. The models corresponding to each property, as well as the *global* classifier are convolutional neural networks with the ResNet50 architecture [22]. The models were pretrained with ImageNet then finetuned with our dataset.

We want to evaluate the capacity of our system to accurately tell when a prediction is correct or incorrect and compare it to the *global* classifier's performance. The correctness of a prediction corresponds to whether the class prediction from the *global* classifier is correct or not. Therefore, we divide the predictions into 4 categories: Correct Consistent (CC), Correct Inconsistent (CI), Incorrect Consistent (IC) and Incorrect Inconsistent (II). A 5-fold cross-validation was performed on the system to compute evaluation metrics. Table 1 shows a few metrics computed for each class then aggregated with the mean value, the maximum and minimum values and the standard deviation to observe the dispersion of the results for each class.

Table 1. Results of the experiment

Metric	Mean	Max	Min	Standard Deviation
Global classifier accuracy	0.80	0.97	0.34	0.19
System accuracy	0.79	0.96	0.40	0.17
False Positive Rate	0.72	0.94	0.35	0.17
False Negative Rate	0.05	0.22	0	0.06

The system accuracy is computed as $\frac{CC+II}{n}$ because we want to evaluate the capacity of the system to flag incorrect predictions as inconsistent and correct predictions as consistent. The system accuracy is similar to the *global* classifier



Bassoon

The model saw a **Bassoon** and it is **not consistent** with the observed properties.

The following properties are inconsistent:

- Not hasMechanism Keys

- Green bar: the system has decided the property is present because it has a probability higher than 70 %.
- Red bar: the system has decided the property is absent because it has a probability lower than 30 %.
- Orange bar: the system could not decide about the presence/absence of this property.

Fig. 1. The explanation interface in the case of an inconsistent prediction

accuracy, which means the impact of the consistency check is minimal on performance. However, the accuracy is biased since a majority of the predictions are correct. That is why we study the False Positive Rate (FPR) and False Negative Rate (FNR), considering IC as false positives and CI as false negatives. The values of FPR and FNR show that the system classifies most predictions as consistent, regardless of the prediction's correctness. This issue may come from two factors.

The first factor is the ontology's design. Since the consistency is checked using the open-world assumption, the predictions are unlikely to be inconsistent if class definitions do not have strong restrictions, such as negative restrictions.

A second factor is the values of the thresholds described in section 3.3. They have a direct impact on FPR and FNR. Indeed, these thresholds decide whether to add a restriction or not. In this experiment, these thresholds were 0.7 and 0.3 for $threshold^{+}$ and $threshold^{-}$ respectively. From our observations, it is clear that the thresholds should be different per model to account for the variation in their performance. This experiment highlighted the challenges to make these consistency checks as accurate as possible.

Let us now focus on the explanation interface. Figure 1 shows the interface for an inconsistent prediction. It is divided into three parts, the input image on the top left, a summary of the prediction on the top right and the outputs of every model on the bottom. As discussed in 3.4, the summary of the prediction contains the list of all the inconsistent restrictions.

On the bottom, there are multiple tabs that each contains a bar chart of the outputs of one model. The figure shows the chart for the property `hasMechanism` as it demonstrates the color coding when there is an inconsistency. The bars are all red, meaning that the outputs are below $threshold^{-}$, therefore negative restrictions are added. Colored words on the left of the graph are the labels or target classes of the selected model. The label `Keys` is in red because the restriction "Not *hasMechanism* `Keys`" is inconsistent. The other labels under `Keys` are green because the negative restrictions using them as target classes are consistent. On the right of the bar chart is a legend to help the user read the chart. This legend shows the thresholds for the studied property.

From this interface, we believe that a user can extract some useful information. Seeing the input image allows the user to assess its quality. This particular image poses a problem to the system, since it cannot handle more than one instrument. The bar charts also help determine the level of confidence of the system. When displaying the output of the *global* classifier, we see that the class `Bassoon` has the highest probability with 35%. From this value, we can hypothesize that the classifier was confused and actually saw different instruments. From this hypothesis, we infer that having multiple instruments in an image is not handled well and therefore is a limitation of the system.

The restriction causing the inconsistency was very close to not being added. Indeed, the probability is of 29% and the threshold is at 30%. This indicates that the restriction may actually be consistent. Adding counterfactual explanations will help detect these cases.

This interface has not been evaluated since it not yet complete. We plan to add new pages that correspond to different types of explanations. We also want to systematically propose three different levels of explanations corresponding to the levels described in section 3.4.

5. Conclusion and future work

This paper introduced a new explainable system that combines ontologies with machine learning models and shows the results in a custom explanation interface. It is a step towards trustworthy and robust predictive systems. Indeed, the capacity of detecting when a prediction is incorrect and warning the user increases the trustworthiness of the system. Likewise, the use of concurrent statistical models whose results are aggregated using logic increase the system's robustness, compared to using a single model to make the prediction. This system also requires minimal additional work to be implemented with regard to building a new ontology and statistical models for a new task. Indeed, any existing ontology can be utilized, only observable properties should be added and used in some class definitions. In a similar fashion, any statistical model architecture can be used, allowing the usage of already trained models instead of training new ones.

The system was evaluated with a musical instruments classification task. The system's capacity of detecting incorrect predictions is not yet satisfying, but it demonstrated the feasibility and should be seen as a proof of concept. The explanation interface shows the main information about the results of a prediction. It is the first element of a broader interface that will include different types and levels of explanations. It is designed with a Model-View-Controller design pattern to make it modular and compatible with other predictive models or other interfaces.

We believe that this system enables the generation of counterfactual explanations. Indeed, it is possible to modify the ontology or the thresholds to see how the system behaves. To fully verify this hypothesis and further explore the explainability of our architecture, we intend to work on how to design counterfactual explanations using an ontology. Specifically, how to find the minimum changes in the ABox of the ontology that changes the outcome of the system. We also plan to further develop our interface, by providing explanations for three levels of expertise: AI experts, domain experts and non-experts. This architecture is also perfectly fit for logical explanations. For instance, the Pellet reasoner [23] provides detailed explanations for inconsistencies that could be used for our system. In section 2 we discussed that there were many toolkits to generate explanations, but they are accessible only to AI experts. The current state of our interface did not solve this issue. That is why in future works, we plan to use these toolkits to generate *raw explanations* that we will then adapt in order to make them understandable to different types of users. We will use existing toolkits to compute different kinds of explanations and adapt them to the chosen level of expertise. Finally, to create an accessible user-friendly interface, we will use the guidelines and recommendations discussed in section 2.

References

[1] V. Arya, R. K. Bellamy, P.-Y. Chen, A. Dhurandhar, M. Hind, S. C. Hoffman, S. Houde, Q. V. Liao, R. Luss, A. Mojsilović, et al., One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques, arXiv preprint arXiv:1909.03012 (2019). arXiv:1909.

03012.

[2] D. Alvarez-Melis, T. S. Jaakkola, Towards robust interpretability with self-explaining neural networks, Advances in neural information processing systems 31 (Jun. 2018). arXiv:1806.07538.

[3] S. Sachan, J.-B. Yang, D.-L. Xu, D. E. Benavides, Y. Li, An explainable AI decision-support-system to automate loan underwriting, Expert Systems with Applications 144 (2020) 113100.

[4] D. Gunning, D. Aha, DARPA's explainable artificial intelligence (XAI) program, AI Magazine 40 (2) (2019) 44–58.

[5] J. Klaise, A. V. Looveren, G. Vacanti, A. Coca, Alibi explain: Algorithms for explaining machine learning models, Journal of Machine Learning Research 22 (181) (2021) 1–7.
URL http://jmlr.org/papers/v22/21-0017.html

[6] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence 1 (5) (2019) 206–215.

[7] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viegas, J. Wilson, The what-if tool: Interactive probing of machine learning models, IEEE Transactions on Visualization and Computer Graphics (2019) 1–1doi:10.1109/tvcg.2019.2934619.

[8] K. Sokol, P. Flach, Explainability fact sheets: a framework for systematic assessment of explainable approaches, in: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, ACM, 2020. doi:10.1145/3351095.3372870.

[9] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, E. Horvitz, Guidelines for human-AI interaction, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, ACM, 2019. doi:10.1145/3290605.3300233.

[10] M. Chromik, A. Butz, Human-XAI interaction: A review and design principles for explanation user interfaces, in: Human-Computer Interaction – INTERACT 2021, Springer International Publishing, 2021, pp. 619–640. doi:10.1007/978-3-030-85616-8_36.

[11] Q. V. Liao, D. Gruen, S. Miller, Questioning the AI: Informing design practices for explainable AI user experiences, in: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, ACM, 2020. doi:10.1145/3313831.3376590.

[12] Z. C. Lipton, The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery., Queue 16 (3) (2018) 31–57.

[13] M. Bellucci, N. Delestre, N. Malandain, C. Zanni-Merk, Towards a terminology for a fully contextualized XAI, Procedia Computer Science 192 (2021) 241–250.

[14] N. Guarino, D. Oberle, S. Staab, What is an ontology?, in: Handbook on ontologies, Springer, 2009, pp. 1–17.

[15] A. Seeliger, M. Pfaff, H. Krcmar, Semantic web technologies for explainable machine learning models: A literature review., PROFILES/SEMEX@ ISWC 2465 (2019) 1–16.

[16] G. Futia, A. Vetrò, On the integration of knowledge graphs into deep learning models for a more comprehensible AI—three challenges for future research, Information 11 (2) (2020) 122.

[17] N. Halliwell, F. Gandon, F. Lecue, User scored evaluation of non-unique explanations for relational graph convolutional network link prediction on knowledge graphs, in: Proc. of the 11th on Knowledge Capture Conf., ACM, 2021.

[18] J. M. Rožanec, D. Mladenić, Semantic xai for contextualized demand forecasting explanations, arXiv preprint arXiv:2104.00452 (Apr. 2021).

[19] W3C, OWL 2 web ontology language document overview, Website, accessed on 15/03/2022 (Dec. 2012).
URL https://www.w3.org/TR/owl2-overview/

[20] W3C, OWL 2 web ontology language manchester syntax, Website, accessed on 15/03/2022 (Dec. 2012).
URL https://www.w3.org/TR/owl2-manchester-syntax/

[21] W3C, OWL 2 web ontology language structural specification and functional-style syntax, Website, accessed on 15/03/2022 (Dec. 2012).
URL https://www.w3.org/TR/owl2-syntax/

[22] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016.

[23] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical owl-dl reasoner, Journal of Web Semantics 5 (2) (2007) 51–53.