Human-understandable and Machine-processable Explanations for Sub-symbolic Predictions

Abdus Salam and **Rolf Schwitter** and **Mehmet A. Orgun**
Macquarie University, Sydney, Australia
{abdus.salam, rolf.schwitter, mehmet.orgun}@mq.edu.au

Abstract

HESIP is a hybrid machine learning system in which a sub-symbolic machine learning component makes a prediction for an image classification and afterwards a symbolic machine learning component learns probabilistic rules that are used to explain that prediction. In this paper, we present an extension to HESIP that generates human-understandable and machine-processable explanations in a controlled natural language for the learned probabilistic rules. In order to achieve this, the literals of the probabilistic rules are first reordered, and then aggregated and disambiguated according to linguistic principles so that the rules can be verbalised with a bi-directional grammar. A human-in-the-loop can modify incorrect explanations and the same bi-directional grammar can be used to process these explanations to improve the decision process of the machine.

1 Introduction

The recent success of machine learning (ML) models is remarkable, especially the success of sub-symbolic ML models in problem domains that are computationally expensive, such as natural language processing and image processing (Zhang et al., 2020; LeCun et al., 2015). Sub-symbolic ML models mostly learn functions that map the input data to the output data to find the correlations between them (Ilkou and Koutraki, 2020). When a ML model selects one or more class labels as an output for a given input, the output is known as a prediction of the model. These sub-symbolic models are used in intelligent systems to make better decisions based on their predictions. Since most of these sub-symbolic ML models are black-box models that are not immediately interpretable, it is difficult to explain to a user of an intelligent system how the machine learning algorithm came to a particular decision. This is why eXplainable AI (XAI) has recently gained momentum, since this discipline aims to produce explainable models that humans can understand, manage and trust (Gunning, 2017).

Most systems that can explain a prediction, such as Lime (Ribeiro et al., 2016) and Anchor (Ribeiro et al., 2018), explain the prediction based on features that exist in the dataset. For example, Lime selects super-pixels to explain an image prediction. This explanation may be helpful for a domain expert, but it may be difficult to understand for a non-expert user. Alternatively, researchers have tried to employ symbolic ML models to explain predictions, since these models are directly interpretable (Rabold et al., 2019). Symbolic ML models learn symbolic rules which are then presented to the user as an explanation for a prediction. Although these symbolic rules are easier to interpret as shown in several studies (Muggleton et al., 2018), they are sometimes difficult to understand by a user who does not have a background in formal methods. Therefore, it is important to study alternative ways of generating explanations that are human-understandable (and as we will argue later at the same time machine-processable).

In this paper, we present a linguistic extension to HESIP, a hybrid explanation system for image prediction. The extended implementation of the system uses a bi-directional grammar to generate explanations in a controlled natural language.

2 HESIP: System Description

HESIP combines sub-symbolic and symbolic representations in two separate components of the system to construct symbolic explanations for image predictions. According to Kautz's classification, HESIP is a hybrid system of Type-3 where a sub-symbolic component is used to work on a task, and then a symbolic component is used to finalise that task (Garcez and Lamb, 2020). HESIP is motivated by LIME-Aleph (Rabold et al., 2019) that is an explanation system where an image prediction is explained from the learned rules. The LIME-

Aleph system relies on two datasets that contain synthetic images. HESIP extends the architecture of the LIME-Aleph system and uses a more generalised approach so that it can be applied to real-world datasets. The architecture of the HESIP system is illustrated in Figure 1.

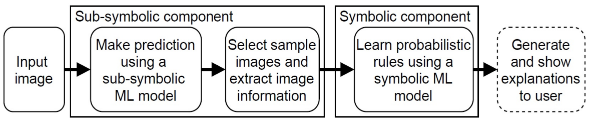
Input image
Sub-symbolic component
Make prediction using a sub-symbolic ML model
Select sample images and extract image information
Symbolic component
Learn probabilistic rules using a symbolic ML model
Generate and show explanations to user

Figure 1: Architecture of HESIP

2.1 The Sub-symbolic Component

HESIP takes an image as input and makes a prediction with a probability using an artificial neural network (ANN) (Krizhevsky et al., 2012) as a sub-symbolic ML model. Based on the similarity to the input image, HESIP selects positive and negative instances of sample images. The probability of each of these sample images is predicted by the ANN. If the prediction probability of a sample image is greater than or equal to the prediction probability of the input image, then the sample image is a positive instance; otherwise, it is a negative instance. HESIP extracts the image information for all the positive and negative image instances. This image information is then used as observed data in the symbolic component of HESIP for learning probabilistic rules that explain a prediction.

Let us illustrate these steps in more detail using a motivating example from our own dataset that contains images of a particular shape. The task is then to determine if an image represents the concept of a house. Each image consists of two objects and each object has either the shape of a square or a triangle. The colour of these objects is either green or blue. We say that an image represents the concept of a house, if the image contains an object of type triangle that is located on the top of an (adjacent) object of type square. In any other case, the image does not represent the concept of a house (see Figure 2).

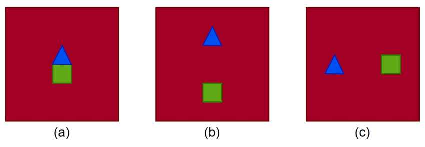

Figure 2: Examples of house concept learning. (a) represents the concept of a house, while (b) and (c) do not.

HESIP extracts the information from the image that will be used for the explanation; for example, information about the location of objects (e.g., position), about their properties (e.g., colour), and the relation between the objects, (e.g., on top of or left of). HESIP employs Detectron2 (Wu et al., 2019), a PyTorch-based object detection library that supports the Mask R-CNN method (He et al., 2017) to detect objects and their location information in an image. The location information is then used to extract the colour of the objects and to determine the relations between two objects in an image.

In our case, two objects can stand in the following relations: *left of*, *right of*, *top of*, *bottom of*, *on*, *under* and *contain*. Note that the relation *on* holds if an object is on the top of another object and these two objects are adjacent; similarly, the relation *under* holds if an object is beneath another object and these two objects are adjacent.

2.2 The Symbolic Component

HESIP employs a probabilistic logic programming framework called *cplint* (Riguzzi and Azzolini, 2020) as the symbolic component to learn probabilistic rules from data about positive and negative instances of sample images. The decision about whether an image is a positive or negative instance is provided by the sub-symbolic component. The probabilistic rules are then used to explain the prediction of an input image. These probabilistic rules have the following abstract form (Vennekens et al., 2004):

h_1:a_1 ;...; h_n:a_n :- b_1,..., b_m.

where h_i are atoms, b_i are literals (incl. negation as failure), and a_i is the probability, a real number between 0 and 1. The set of elements h_i:a_i forms the head of a rule and the set of elements b_i the body; the head and the body are separated by an *if*-symbol (:-). A disjunction in the head of a rule is represented with a semicolon (;) and atoms are separated by a colon (:) from probabilities.

Once the sample image information is available, HESIP represents the image using a simple ontology so that we can also generate explanations for other application domains using the same method. The ontology employed in HESIP consists of four predicates: (1) `object/1` represents an object; (2) `type/2` represents an object type; (3) `property/3` represents an object property; and (4) `relation/3` represents a relation between two objects. The probabilistic rules in HESIP contain only those predicates (atoms and literals) that are available in the ontology. In our case, the head of a rule may contain the predicate `relation/3` or `type/2` and the body may contain any predicate of the ontology.

In our context, HESIP generates a probabilistic rule as shown in Listing 1 which states that an object A is of type house if all the conditions in the body of the rule are satisfied and the probability is `1`. Once a probabilistic rule is available, HESIP verbalises that rule in a controlled natural language (CNL) (Kuhn, 2014) and displays it together with the corresponding probability to explain the prediction of an input image.

Listing 1: A sample rule for house concept learning

```
type(A, house) : 1.0 :-
    type(B, triangle),
    object(B),
    type(C, square),
    object(C),
    relation(B, C, on),
    property(C, green, colour),
    property(B, blue, colour),
    relation(A, C, contain),
    relation(A, B, contain),
    object(A).
```

3 Generating Explanations

The rule in Listing 1 cannot be immediately verbalised, since the literals are not in an order that follows a linguistically motivated pattern. Our goal is to generate an explanation of the following form:

> *If an object contains a blue object of type triangle and contains a green object of type square and the blue object is located on the green object then the object is of type house.*

In order to achieve this, we use a bi-directional definite clause grammar similar to the one proposed by Schwitter (2018) that takes a set of reconstructed rules as input and generates explanations in a CNL as output. The same bi-directional grammar can be used to process a (modified) explanation and translate it into a rule as long as we stick to the syntax of the CNL. That means the CNL serves as a high-level interface language to the HESIP system and the user can modify and refine explanations and feed them back to the system.

3.1 Order of Content

Before we can verbalise the content of a rule, we need to identify those literals that introduce new content and distinguish them from literals that link to previously introduced content, and then reorder these literals according to a linguistic pattern. This process leads to a reconstruction of the rule where some literals are repeated so that they correspond to the underlying linguistic pattern. After reordering, the reconstructed rule looks as shown in Listing 2:

Listing 2: A sample rule after reconstruction

```
class(A, object), type(A, house) :-
    class(A, object),
    relation(A, B, contain),
    property(B, blue, colour),
    class(B, object),
    type(B, triangle),

    class(A, object),
    relation(A, C, contain),
    property(C, green, colour),
    class(C, object),
    type(C, square),

    property(B, blue, colour),
    class(B, object),
    type(B, triangle),
    relation(B, C, on),
    property(C, green, colour),
    class(C, object),
    type(C, square).
```

The body of this rule consists of three implicit linguistic patterns. The reordered literals in these patterns follow now a subject-verb-complement structure. The first two patterns use the same sequence of literal types and introduce new content; the subject position is occupied by a class, the verb position by a relation, and the complement position by a property, followed by a class and a type. The third pattern only uses new content in the verb position but previously introduced content in the subject and complement positions. The head of the rule consists of a pattern with a similar structure of the form subject-copula-complement where the subject position holds a class, the copula position is not filled, and the complement position holds a type. For this reconstructed rule, our grammar generates the following verbalisation:

> *If an object contains a blue object of type triangle and the object contains a green object of type square and the blue object of type triangle is located on the green object of type square then the object is of type house.*

This verbalisation is very explicit and can be improved using a number of micro-planning strategies (Reiter and Dale, 2000). However, since our goal is to generate explanations that are human-understandable as well as machine-processable, we need to make sure that we do not introduce any ambiguities during micro-planning.

3.2 Aggregation of Content

Aggregation is the process of removing redundant information in a sentence (Dalianis and Hovy, 1993). In our case, we can use subject grouping to combine clauses and drop type information that has already been introduced to reduce redundancy (see Listing 3).

Listing 3: A sample rule after performing aggregation

```
class(A, object), type(A, house) :-
    class(A, object),
    relation(A, B, contain),
    property(B, blue, colour),
    class(B, object),
    type(B, triangle),
    relation(A, C, contain),
    property(C, green, colour),
    class(C, object),
    type(C, square),

    property(B, blue, colour),
    class(B, object),
    relation(B, C, on),
    property(C, green, colour),
    class(C, object).
```

Subject grouping results in verb phrase coordination and removing type information results in more compact definite descriptions as shown in our target explanation at the beginning of Section 3. Note that reprocessing of this explanation by our bi-directional grammar results in a semantically equivalent rule.

3.3 Generating Definite Descriptions

During generation, the bi-directional grammar stores all the accessible antecedents and generates minimal definite descriptions on the fly. However, the grammar would generate an ambiguous verbalisation if we had a rule where the object in the head does not have a unique object in the body to link to after reconstruction and aggregation. To avoid this kind of an ambiguity, we add a variable to such an underspecified rule that allows us to distinguish between objects in an explicit way on the surface level of an explanation. For the learning of a house concept, this kind of an ambiguity occurs if we do not use type information for objects. Therefore, we add a variable to resolve the ambiguity (see Listing 4).

Now the following unambiguous verbalisation can be generated that introduces an indefinite noun phrase with a variable *an object A* as antecedent for the definite description *the object A* that occurs in the consequent of the sentence:

> *If an object A contains a blue object and contains a green object and the blue object is located on the green object then the object A is of type house.*

Listing 4: A sample rule after adding variables

```
class(A, object), variable(A, 'A'),
type(A, house) :-
    class(A, object),
    variable(A, 'A'), ...
```

4 Evaluation

We evaluate our explanation generation method in two ways: (1) we check if a generated explanation corresponds to a minimal and correct description of the image information, and (2) we check if the bi-directional grammar correctly works in both directions. For the first evaluation, we check whether an explanation is minimal or not by testing if the explanation meets our aggregation criteria. We check the correctness of an explanation by matching the literals used for the verbalisation with the corresponding literals for the image. For the second evaluation, we check the bi-directional grammar via a technique that is known as semantic round-tripping (Hossain and Schwitter, 2020); basically, we keep the formal representation *R1* for an explanation, feed that explanation again to the bi-directional grammar, generate a formal representation *R2*, and then compare if *R1* and *R2* are semantically equivalent.

In addition to our house concept dataset, we have employed the tower concept dataset and used single relation learning to evaluate our explanation generation method. This additional dataset is also used in the LIME-Aleph system to illustrate their method. For the learning of a tower concept, an image consists of three square objects of different colours and we say that the image represents a tower concept if one square is on the top of another square without the repetition of objects with the same colour. For single relation learning, we say that an image represents the *left of* relation if a green square is on the left side of a blue square. We used 1000 images from each dataset for the evaluation. For each image, the explanation is generated and evaluated using the above-mentioned technique. We have found that all the explanations for tower concept and for single relation learning are correct while for house concept learning 999 explanations are correct. This gives us an accuracy of 100%, 100% and 99.9%, respectively.

5 Conclusion

In this paper, we have introduced a linguistic extension to HESIP, a hybrid explanation system, that combines sub-symbolic and symbolic representations for image predictions. This linguistic extension uses a bi-directional logic grammar to generate explanations in a CNL. The sub-symbolic component of HESIP makes a prediction that results in a probabilistic rule in the symbolic component of the system. The resulting rule is reordered according to linguistic principles, redundant information is aggregated, and possible ambiguities are resolved, before the rule is processed by the grammar. The output of the grammar is an unambiguous explanation of the prediction. If this explanation is not correct, then the user can modify the explanation and feed it back to HESIP.

The advantage of HESIP over the LIME-Aleph system is that it employs an object detection model to find objects in the images and uses an ontology to represent image information. The novelty of our hybrid approach to machine learning is that it allows us to generate explanations that are human-understandable as well as machine-processable; and it can be customised for other prediction tasks. HESIP can be used in any real-world application of image prediction where the images in the dataset have relations between objects. These relations can then be used in the probabilistic rules to explain the image predictions. Currently, we are investigating how HESIP can be extended and used to learn concepts from different parts of objects using the PASCAL-Part (Chen et al., 2014) dataset.

References

Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. 2014. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proc. CVPR'14*, pages 1971–1978.

Hercules Dalianis and Eduard Hovy. 1993. Aggregation in natural language generation. In *EWNLG'93*, pages 88–105. Springer.

Artur d'Avila Garcez and Luis C. Lamb. 2020. Neurosymbolic AI: The 3rd Wave. *arXiv preprint arXiv:2012.05876*.

David Gunning. 2017. Explainable artificial intelligence (XAI). *Defense Advanced Research Projects Agency (DARPA)*.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In *IEEE ICCV'17*, pages 2980–2988.

Bayzid Ashik Hossain and Rolf Schwitter. 2020. Semantic round-tripping in conceptual modelling using restricted natural language. In *Australasian Database Conference*, pages 3–15. Springer.

Eleni Ilkou and Maria Koutraki. 2020. Symbolic Vs Sub-symbolic AI Methods: Friends or Enemies? In *CIKM (Workshops)*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS'12*, pages 1097–1105.

Tobias Kuhn. 2014. A survey and classification of controlled natural languages. *Computational Linguistics*, 40(1):121–170.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436.

Stephen H. Muggleton, Ute Schmid, Christina Zeller, Alireza Tamaddoni-Nezhad, and Tarek Besold. 2018. Ultra-Strong Machine Learning: comprehensibility of programs learned with ILP. *Machine Learning*, 107(7):1119–1140.

Johannes Rabold, Hannah Deininger, Michael Siebers, and Ute Schmid. 2019. Enriching visual with verbal explanations for relational concepts–combining LIME with Aleph. In *ECML PKDD'19*, pages 180–192. Springer.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *ACM SIGKDD ICKDD'16*, pages 1135–1144. ACM.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. *In AAAI'18*, 32(1).

Fabrizio Riguzzi and Damiano Azzolini. 2020. cplint Manual. *SWI-Prolog Version*. Retrieved May 14, 2021 from `http://friguzzi.github.io/cplint/_build/latex/cplint.pdf`.

Rolf Schwitter. 2018. Specifying and verbalising answer set programs in controlled natural language. *Theory and Practice of Logic Programming*, 18(3-4):691–705.

Joost Vennekens, Sofie Verbaeten, and Maurice Bruynooghe. 2004. Logic programs with annotated disjunctions. In *ICLP'04*, pages 431–445. Springer.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. Retrieved May 14, 2021 from `https://github.com/facebookresearch/detectron2`.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep Learning on Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering*. DOI: 10.1109/TKDE.2020.2981333.