Generating and Modifying Natural Language Explanations

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 explanation system for image predictions that combines sub-symbolic and symbolic machine learning techniques to explain the predictions of image classification tasks. The sub-symbolic component makes a prediction for an image and the symbolic component learns probabilistic symbolic rules in order to explain that prediction. In HESIP, the explanations are generated in controlled natural language from the learned probabilistic rules using a bi-directional logic grammar. In this paper, we present an explanation modification method where a human-in-the-loop can modify an incorrect explanation generated by the HESIP system and afterwards, the modified explanation is used by the symbolic component of HESIP to learn a better explanation.

1 Introduction

In recent years, the development of explanation systems has gained a lot of attention. Most of these explanation systems (Ribeiro et al., 2016, 2018; Lundberg and Lee, 2017) can explain predictions made by machine learning (ML) models. Researchers are focusing on building explanation systems for ML models, because these models have shown excellent performance for different prediction tasks (Zhang et al., 2020; LeCun et al., 2015) and most of these models are sub-symbolic black-box models that are not easily understandable, and therefore lead to difficulties explaining the predictions to the users. Explanation systems such as Lime (Ribeiro et al., 2016), Anchor (Ribeiro et al., 2018) and SHAP (Lundberg and Lee, 2017) use existing information of the datasets to explain predictions. However, sometime information that is not present directly in the dataset such as relation information can play an important role in the explanation; especially, in image prediction tasks as shown in LIME-Aleph (Rabold et al., 2019).

HESIP is a hybrid explanation system for image predictions. The HESIP system explains the predictions to the users using natural language explanations. The explanations are generated in a controlled natural language (CNL) (Kuhn, 2014) using a logic programming based bi-directional grammar that is similar to Schwitter (2018). The generated explanations of the HESIP system are human-understandable as well as machine-processable. Since the explanations are represented in a natural language, they are immediately understandable by all types of users. The bi-directional grammar of the HESIP system can also process a generated explanation that has been modified by the user. The HESIP system aims to generate an explanation for the predicted image that represents the object information together with the relation information. It is expected that such as system is not perfect, and HESIP is not an exception. HESIP sometimes generates wrong explanations. To the best of our knowledge, there is no explanation system that allows a user to modify an explanation in order to improve the explanation generation process of the system. It is important that a user can modify an incorrect explanation so that the system can learn how to generate a better explanation taking the feedback from the user into consideration. In this paper, we present a method that involves a human-in-the-loop who can fix incorrect explanations by modifying them.

2 HESIP: System Architecture

HESIP is a hybrid system that explains image predictions by integrating sub-symbolic and symbolic ML techniques in two separate components. For an input image, HESIP makes a prediction using a sub-symbolic ML model. Afterwards, HESIP uses a symbolic ML technique to learn symbolic probabilistic rules that are used to explain predic-

tions. Based on a definition of hybrid systems introduced in Kautz's classification (Kautz, 2020), the HESIP system follows the architecture of a Type-3 hybrid system, since HESIP uses a sub-symbolic component to work on a task, and then a symbolic component to finalise that task. Figure 1 shows the architecture of the HESIP system. More details about the HESIP system can be found in (Salam et al., 2021).

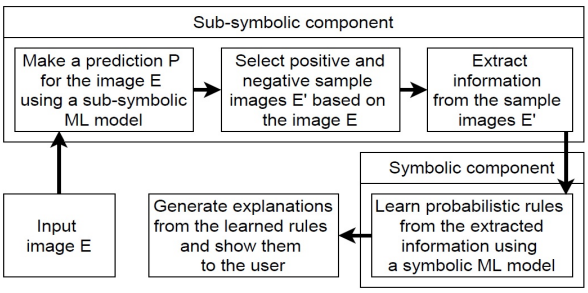

Figure 1: Architecture of the HESIP system

Rabold et al. (2019) have developed an explanation system called LIME-Aleph that explains image predictions using the learned rules. The HESIP system is motivated by the LIME-Aleph system and extends the architecture of LIME-Aleph in order to achieve a more generalised method. LIME-Aleph depends on two datasets that consist of synthetic images while the HESIP system can be applied to datasets that consist of real-world images. A detail comparison between the HESIP system and the LIME-Aleph system is provided in (Salam et al., 2021).

The steps of the HESIP system are demonstrated here using the PASCAL-Part dataset (Chen et al., 2014) that consists of real-world images. We want to learn the concept of a potted plant from the different parts of the concept that are present in an image. For this learning task, only those images that contain potted plants and bottles are used from the dataset. The potted plant concept has two parts: pot and plant. We say that there is a potted plant concept in an image, if it represents a pot that is located below a plant. Similarly, the bottle concept consists of two parts: body and cap. There are images of bottles in the dataset that contain only a body part. In our case, we work with the bottle images that contain both parts. Figure 2 shows images that contain a potted plant and a bottle.

2.1 The Sub-symbolic Component

As sub-symbolic ML model, HESIP uses an artificial neural network (ANN) (see Russell and Norvig, 2020, for an introduction) to make a prediction with a probability for an input image. Therefore, positive and negative images are selected from the dataset in order to learn explanatory probabilistic rules in the symbolic component. The sample images are selected based on the similarity to the input image. Predictions with their probabilities for all sample images are made with the ANN. A sample image is considered as a positive instance, if the prediction probability of the input image is less than or equal to the prediction probability of the sample image; otherwise, the sample image is considered as a negative instance.

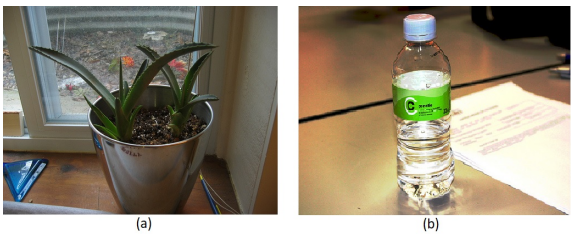

Figure 2: Example of a potted plant (a) and a bottle (b).

Once the sample images are selected, the HESIP system extracts all image information and represents it using an ontology. After that, the symbolic component uses this image information as data to learn the probabilistic rules. In the image information extraction step, the objects present in the image and their property information, the location information of the objects and the relations between the objects are extracted. The location of an object is determined from its position in the image grid considering the image as a grid. For an image, HESIP detects the objects and their location information using Detectron2 (Wu et al., 2019) that implements the Mask R-CNN (He et al., 2017) object detection algorithm. The relations between the objects in an image are determined using the location information of the objects. We assume that two objects are related in any of the following ways: *left of*, *right of*, *top of*, *bottom of*, *on*, *under* and *contain*. The relation between two objects is *on* or *under*, if one object is at the top or at the bottom of another object and they are adjacent.

2.2 The Symbolic Component

The symbolic component of the HESIP system learns the explanatory rules using the sample image information. As a symbolic component, HESIP uses *cplint* that is a probabilistic logic programming framework (Riguzzi and Azzolini, 2020). The in-

formation about the positive and negative sample images are used as data in the symbolic component that learns probabilistic rules and the predictions of the images are then explained using these rules. A probabilistic rule has the following form:

`h:p :- b1,..., bn.`

where `h` is a head literal, `b1,..., bn` are body literals and `p` is a real number between `0` and `1` that indicates the probability of the rule. The *if*-symbol (`:-`) separates the head and the body of the rule. A colon (`:`) is used to associate the probability with the literal in the head of the rule.

To represent the sample image information, the HESIP system uses an ontology that has four predicates: `object/1`, `type/2`, `property/3` and `relation/3`. The predicates `object/1`, `type/2` and `property/3` are used to represent an object, the type of the object and the property of the object. The relation between two objects is presented using the predicate `relation/3`. The probabilistic rules learned in the HESIP system may contain either the predicate `type/2` or `relation/3` in the head of the rule and may contain any predicates of the ontology in the body of the rule. This ontology makes sure that the explanation generation method used in the HESIP system can be applied to different application domains.

Once the information of the sample images is represented with the help of the ontology, it can be used as data in the symbolic component where the information about each image represents an example instance. The decision of the sub-symbolic component is used to determine whether an example instance is a positive or negative instance. In our case, the images of the potted plant concept are determined as positive example instances while the images of the bottle concept are determined as negative example instances. Listing 1 shows a positive example instance for the potted plant concept in the symbolic component.

Listing 1: A positive example instance for the potted plant concept.

```
begin(model(pp1)).
  object(pp1_obj1).
  object(pp1_obj2).
  object(pp1_obj3).
  type(pp1_obj1, potted_plant).
  type(pp1_obj2, pot).
  type(pp1_obj3, plant).
  relation(pp1_obj1, pp1_obj2, contain).
  relation(pp1_obj1, pp1_obj3, contain).
  relation(pp1_obj2, pp1_obj3, under).
end(model(pp1)).
```

Using these example instances, the symbolic component of the HESIP system learns the probabilistic rule in Listing 2 for the potted plant concept. This rule specifies that an object A is of type potted plant with the probability `1`, if all the literals in the body of the rule are satisfied.

Listing 2: An example of a learned rule for the potted plant concept.

```
type(A, potted_plant):1.0 :-
  type(B, pot), object(B),
  type(C, plant), object(C),
  relation(B, C, under),
  relation(A, C, contain),
  relation(A, B, contain),
  object(A).
```

After the rule is learned in the symbolic component, the HESIP system uses this rule in the explanation generation module in order to generate a natural language description that will explain the image prediction. Before we go into details how this is done and how an explanation can be modified, we first present an overview of the user interface of the HESIP system in the following section.

3 HESIP: User Interface

A prototype of a graphical user interface for the HESIP system has been developed to illustrate the interaction between a user and the system for generating and modifying explanations. As illustrated in Figure 3, a user clicks on the "Choose File" button to select an image for predicting the image. After selecting the image, it is displayed and a new "Predict & Explain" button appears. When the user presses on the "Predict & Explain" button, the HESIP system predicts the image in the sub-symbolic component and learns a probabilistic rule in the symbolic component to explain the prediction. The prediction for the image and the explanation of the prediction along with the probability of the explanation are displayed in a panel (see Figure 3).

Because the interface of the HESIP system displays the predicted image, the prediction and the explanation together, the user can relate and inspect them immediately and can see whether the explanation is correct or not. After showing the prediction and the explanation to the user, two buttons "Modify Explanation" and "Confirm Explanation" are displayed (see Figure 3). After inspection, the user can either modify or confirm the explanation. If the user feels that there is something wrong with

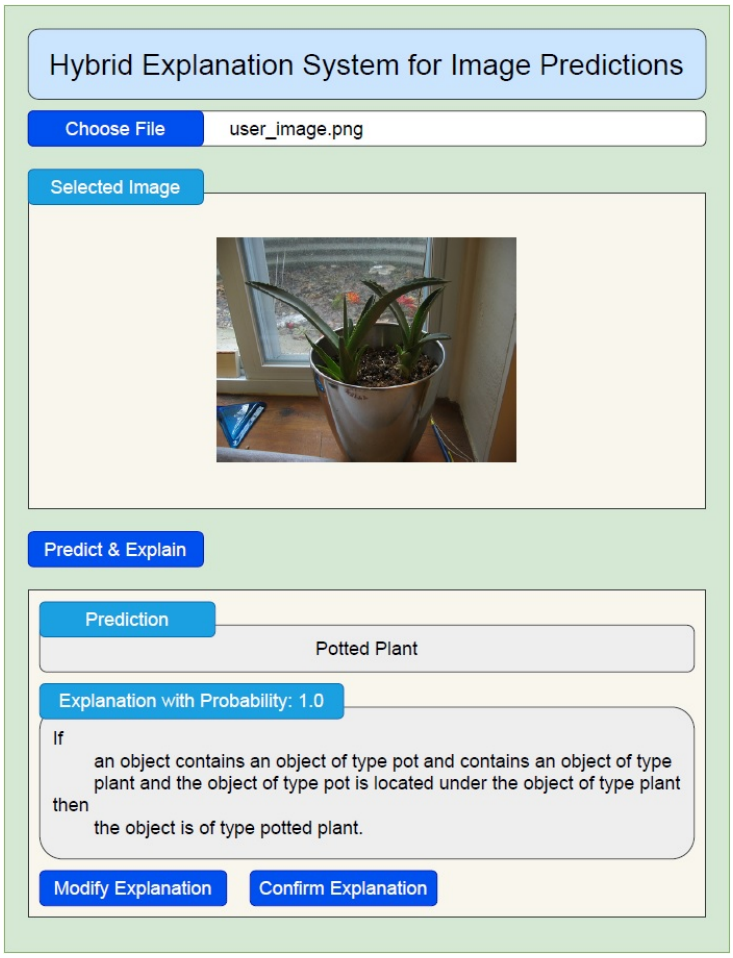

Figure 3: The HESIP system is displaying the prediction of the selected image and the corresponding explanation together with the probability.

the generated explanation, then they can fix the incorrect information so that the HESIP system can learn a better one.

After pressing the "Modify Explanation" button, the HESIP system shows the explanation inside a text editor that allows one to modify the explanation generated by HESIP. When the user uses the text editor to modify the explanation, the editor guides the user using appropriate word suggestions according to the grammar of the CNL used for generating the explanation (see Figure 4). There exist several projects where predictive editors have been developed to guide users for writing sentences in a CNL (Guy and Schwitter, 2017; Franconi et al., 2011; Bernstein and Kaufmann, 2006; Schwitter et al., 2003). When writing a CNL sentence, the next word is predicted and suggested to the user by the predictive editor. Since the explanations are expressed in a CNL, the text editor of the HESIP system can be developed as a predictive editor similar to the PENG^{ASP} system. The user can select a word from the suggested list of words or can write the word manually.

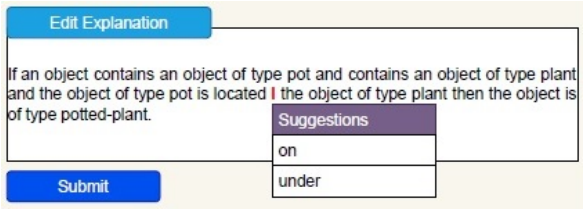

Figure 4: The HESIP system is showing suggestions to the user during the modification of an explanation.

The altered parts of the explanation are displayed as coloured text so that the user can easily identify which parts of the explanation are altered. The user clicks on the "Submit" button once they are done with the modification. Afterwards, HESIP learns a new explanatory rule for the predicted image by taking the modified explanation into account (see Section 5). The modified explanation is first processed using the bi-directional grammar that

Listing 4: Definite Clause Grammar Rule of the Bi-directional Grammar

```
np([mode:M, ctx:body, fcn:subj, def:_D, num:N, arg:X, clause:C1-C5, ante:A1-A4]) -->
  det([mode:M, morph:O, num:N, def:D, clause:C1-C2]),
  noun([mode:M, morph:_, num:N, arg:X, clause:C2-C3, ante:A1-A2]),
  prep([mode:M, ctx:head]),
  rnoun([mode:M, morph:_, num:N, arg:X, clause:C3-C4, ante:A2-A3]),
  { anaphora_resolution(det_noun_prep_rnoun, [M, D, X, C1, C4, C5, A1, A3, A4]) }.
```

produces a rule. Therefore, this rule is used to obtain the modified information and the example instances are updated accordingly in the symbolic component. Finally, the symbolic component uses these updated example instances to learn a new explanatory rule. HESIP generates a new explanation from the newly learned rule and displays it on the interface. The HESIP system compares the previous and the new explanation to identify the differences between them. If any difference was found, then the HESIP system shows that part as coloured text in order to highlight what is different with respect to the previous explanation. The message "*New explanation has been learned using the feedback.*" is displayed along with the new explanation to assure the user that the explanation has been learned taking the user's modified explanation into consideration. At this point, the user can press the "Confirm Explanation" button to approve the new explanation; otherwise, they can make further modifications to the explanation.

4 Generating Explanations

In Section 2, we showed how the HESIP system learns symbolic representation for generating explanations. Now let us have a closer look at how these explanations are generated. Once the explanatory rule is learned in the symbolic component, the HESIP system generates an explanation for the image prediction from the learned rule using a bi-directional logic grammar. The generated explanation can be processed with the same bi-directional grammar to produce a rule that is semantically equivalent to the learned rule from which the explanation was generated. This is important for the modification process, since we want to make sure that the grammar produces correct rules after processing the generated explanations as we will see in the following section.

The learned rule needs to be pre-processed before it can be used by the grammar for generating a natural language explanation. In the pre-processing steps, the literals of the rule are first reordered in a linguistically-motivated way; therefore, subject grouping is applied to remove redundant information in the reordered rule; and finally, variables that serve as names are added to the rule if required in order to resolve ambiguity of definite description. After pre-processing, HESIP sends the reconstructed rule to the grammar that generates the explanation. Listing 3 shows a reconstructed rule for the learned rule (see Listing 2) of the potted plant concept.

Listing 3: A reconstructed rule for the potted plant concept after pre-processing.

```
class(A, object), type(A, potted_plant) :-
  class(A, object),
  relation(A, B, contain),
  class(B, object), type(B, pot),
  relation(A, C, contain),
  class(C, object), type(C, plant),

  class(B, object), type(B, pot),
  relation(B, C, under),
  class(C, object), type(C, plant).
```

For the reconstructed rule in Listing 3, the grammar of the HESIP system generates the following explanation: *If an object contains an object of type pot and contains an object of type plant and the object of type pot is located under the object of type plant then the object is of type potted plant.*

Listing 4 shows an example of a grammar rule that generates a noun phrase in the subject position for a clause pattern that occurs in the body of a rule. In the generation mode (`mode:gen`), this grammar rule takes a class and a type (for example, `class(B, object)` and `type(B, pot)`) as input and generates an indefinite noun phrase (*an object of type pot*) or a definite noun phrase (*the object of type pot*) as output. The argument `clause` holds a difference list (`C1-C5`) with the incoming and outgoing literals. The argument `ante` holds a difference list (`A1-A4`) with the incoming and outgoing accessible antecedents. The call to `anaphora_resolution/2` updates these two difference lists. It is important to note that exactly the same grammar rule can also be used in the processing mode (`mode:proc`), since

the grammar is bi-directional. In the processing mode, the grammar rule takes the generated verbalisation as input and produces a rule as output that is semantically equivalent to the rule from which the verbalisation was generated.

5 Modifying Explanations

The bi-directional property of the grammar enables the HESIP system to modify the generated explanations. In Section 3, we have shown how a human-in-the-loop can alter explanations that are displayed to the user for explaining image predictions. In this section, we show the steps (see Figure 5) that are performed by the HESIP system in order to generate a new explanation from a newly learned rule for the predicted image after modification by the user. Note that the explanation modification steps are completed in the symbolic component of the HESIP system.

Let us assume that a user wants to see the prediction and the explanation for an image shown in Figure 2a that represents a potted plant concept. As discussed in Section 2, HESIP selects sample images for a predicted image, extracts information of the sample images, represents the sample image information using an ontology. Finally, HESIP uses the information of the sample images as example instances in the symbolic component to learn the explanatory rule for explaining the image prediction. Let us assume that HESIP uses the example instances shown in Listing 5 and learns the explanatory rule in Listing 6 for explaining the prediction of the image in Figure 2a. In this case, HESIP generates the explanation "*If an object contains an object of type pot and contains an object of type plant and the object of type pot is located on the object of type plant then the object is of type potted plant.*" from the learned rule in Listing 6 using the bi-directional grammar after applying the pre-processing steps (as discussed in Section 4).

Listing 5: Three example instances that are used to learn the explanatory rule for the potted plant concept.

```
begin(model(pp1)).
  object(pp1_obj1).
  object(pp1_obj2).
  object(pp1_obj3).
  type(pp1_obj1, potted_plant).
  type(pp1_obj2, pot).
  type(pp1_obj3, plant).
  relation(pp1_obj1, pp1_obj2, contain).
  relation(pp1_obj1, pp1_obj3, contain).
  relation(pp1_obj2, pp1_obj3, on).
end(model(pp1)).

begin(model(pp2)).
  object(pp2_obj1).
  object(pp2_obj2).
  object(pp2_obj3).
  neg(type(pp2_obj1, potted_plant)).
  type(pp2_obj1, bottle).
  type(pp2_obj2, body).
  type(pp2_obj3, cap).
  relation(pp2_obj1, pp2_obj2, contain).
  relation(pp2_obj1, pp2_obj3, contain).
  relation(pp2_obj2, pp2_obj3, under).
end(model(pp2)).

begin(model(pp3)).
  object(pp3_obj1).
  object(pp3_obj2).
  object(pp3_obj3).
  neg(type(pp3_obj1, potted_plant)).
  type(pp3_obj2, pot).
  type(pp3_obj3, plant).
end(model(pp3)).
```

Listing 6: A learned explanatory rule for the potted plant concept.

```
type(A, potted_plant):1.0 :-
  type(B, pot), object(B),
  type(C, plant), object(C),
  relation(B, C, on),
  relation(A, C, contain),
  relation(A, B, contain),
  object(A).
```

When this explanation is displayed to a user, then the user may want to modify the explanation after noticing that the relation *on* between the pot and the plant objects is not correct. Let us assume, the user has changed the explanation to "*If an object contains an object of type pot and contains an object of type plant and the object of type pot is located* ***under*** *the object of type plant then the object is of type potted plant.*" where the preposition *on* is replaced by *under*. After submission of the modified explanation, HESIP processes the explanation using the bi-directional grammar to obtain a rule. The generated rule for the modified explanation is shown in Listing 7.

Listing 7: A rule obtained using the bi-directional grammar by processing the modified explanation for the potted plant concept.

```
type(C, potted_plant) :-
  class(C, object),
  relation(C, A, contain),
  class(A, object),
  type(A, pot),
  relation(C, B, contain),
  class(B, object),
  type(B, plant),
  relation(A, B, under).
```

The rule (see Listing 7) derived from the altered explanation is then compared with the rule previ-

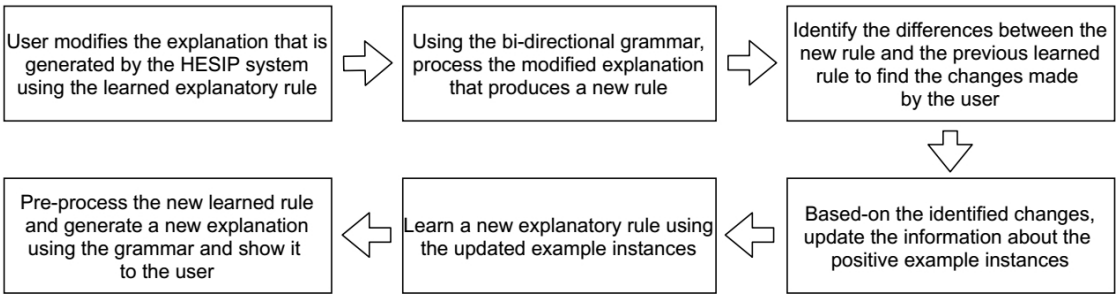

Figure 5: Steps for modifying explanations in the HESIP system.

ously learned by HESIP (see Listing 6) to identify the modifications. In this case, the user has updated the relation information and the rule in Listing 7 reflects that change. After identifying the changes, the amended information is updated in the example instances. In our case, we update the relation from *on* to *under* between the pot and the plant objects for all positive example instances. We do not update any information about the negative example instances, since the user modified an explanation in order to correct it and the positive example instances represent the correct information for the concept to be learned. Afterwards, an explanatory rule is learned using the updated example instances. The new rule is shown in Listing 8. We can see that the new rule is different from the previous one (see Listing 6) and that the preposition has been replaced.

Listing 8: A new explanatory rule learned for the potted plant concept after the explanation is modified by the user.

```
type(A, potted_plant):1.0 :-
  type(B, pot), object(B),
  type(C, plant), object(C),
  relation(B, C, under),
  relation(A, C, contain),
  relation(A, B, contain),
  object(A).
```

Once the new explanatory rule is learned, HESIP first pre-processes the new learned rule as discussed in Section 4 that results in a reconstructed rule. After that, HESIP verbalises the reconstructed rule to obtain a new explanation for the prediction. For this scenario, HESIP generates the new explanation "*If an object contains an object of type pot and contains an object of type plant and the object of type pot is located under the object of type plant then the object is of type potted plant.*" using the new learned rule in Listing 8. The new explanation is then displayed on the interface.

This example illustrates the explanation modification steps for changing the relation information. The HESIP system applies the same process to generate a new explanation for updating any information in the explanation. As mentioned earlier, a predictive editor is used in the HESIP system to modify an explanation that supports the user in making a modification. After generating an explanation for a prediction, it is possible that the explanation may have the following incorrect information and a user can update that information in the explanation:

- The user can update the relation information of an explanation as shown for the explanation of the potted plant concept.
- The user can modify the object property information (for example, the object colour information of a concept).
- The user can update the object type information in the conditional part of the explanation sentence.

Practically, in an explanation sentence, the user can update any content word introduced by the ontology used in the system. In the modification step, the predictive editor will ensure that the explanation is grammatically correct. In the case of updating the relation information, there should not be any issue, since the relevant relation words will be suggested by the predictive editor and the user can select the relation from a list of words. However, a problem may occur while updating any object property or type information in an explanation. An explanation may contain an anaphoric reference to an object. If the user updates the property or the type information of an object that is used as an anaphoric expression, then the user has to make sure that all the other parts of the explanation are also updated. Let us consider an example from a tower concept learning task (Rabold et al., 2019)

to illustrate this scenario. In tower concept learning, an image consists of three squares with green, blue and pink colours, and if the image contains a square on top of another square without repeating the same colour, then the image represents a tower concept. For the tower concept, the HESIP system may generate the following explanation: "*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 tower.*" where the user may update the colour *green* to *pink* only for the first occurrence (*a green object* to *a pink object*). This will lead to incorrect information in the later part of the explanation (the noun phrase *the green object* should be changed to *the pink object*). The predictive editor usually will not identify this information as incorrect, since the sentence is grammatically correct. One possible solution to overcome this problem is to design the predictive editor in such a way that whenever a word related to an anaphoric expression will be updated, the editor will highlight all relevant anaphoric expressions in the explanation and the user can then fix the relevant words.

6 Evaluation

We evaluate the explanation generation and modification process of the HESIP system using four datasets: potted plant concept learning, house concept learning, tower concept learning and single relation learning. The LIME-Aleph system has used the tower concept learning and the single relation learning tasks in order to demonstrate their method. For the house concept learning task, an image consists of a triangle and a square, and if the image contains a triangle on top of a square, then the image represents a house concept. For single relation learning, if an image contains a green square on the left side of a blue square, then the image represents the *left of* relation. For evaluation, we use 382 test images for the potted plant concept and 1000 test images for all other concepts.

The explanation generation process is evaluated in two ways. First, if the generated explanation represents the literals that correspond to the literals of the image, then we consider the explanation as a correct one. Second, we check if the bi-directional grammar works in both directions using a technique known as semantic-round tripping (Hossain and Schwitter, 2020). Using this technique, we store the formal representation *R1* of an explanation. The explanation is then processed by the grammar that produces second formal representation *R2*. Therefore, we compare if *R1* and *R2* are semantically equivalent. For the 1000 test images, HESIP generates all correct explanations for single relation learning and tower concept learning, and 999 correct explanations for house concept learning leading to the accuracy of 100%, 100% and 99.9%. For the 382 test images of potted plant concept, HESIP generates 310 correct explanations with an accuracy of 81.15%.

To evaluate the explanation modification process, we take the test images for which the HESIP system could not generate the correct explanations. HESIP could not generated correct explanations for 1 test image of the house concept and for 72 test images of the potted plant concept learning. To check if the modification process works correctly, we first modify an incorrect explanation; therefore, HESIP learns a new explanatory rule taking the modified explanation into consideration and finally, we check if the explanation generated from the newly learned rule is correct or not.

Among the 72 test images of the potted plant concept for which HESIP could not generate the correct explanations, it generated 53 explanations with wrong relations. We discussed one such explanation in Section 5 where the explanation is learned for the relation *on* instead of the relation *under*. For all 53 explanations, we modified the explanations with correct relations and follow the steps discussed in Section 5 to generate the new explanations. We found that the HESIP system generated the new explanations with correct relations for all of them. For two test images, we could not modify the explanation using a correct relation, since the images contain only plants and there was no pot in these images. We also observe that there were 17 test images of potted plants for which the HESIP system learned incomplete explanatory rules and as a result, the system could not generate suitable explanations that can be modified to generate the correct explanations. We also notice a similar problem for one test image of the house concept for which the HESIP system could not generate a suitable explanation.

7 Conclusion

In this paper, we presented an explanation modification method for HESIP, a hybrid explanation system for image predictions. Using the prototype

of the HESIP system, we showed how a human working in the loop can fix an incorrect explanation generated by the system. For an image prediction, the HESIP system learns an explanatory rule and generates an explanation in CNL using a bi-directional logic grammar. If the user decides that the generated explanation is wrong, then they can modify the explanation to fix it following the grammar rules of the CNL. After modifying the explanation, HESIP learns a new explanatory rule taking the user's modified explanation into account and generates an explanation from the new learned rule to better explain the image prediction. The result of the evaluation shows that the modification process of the HESIP system is very effective in learning better explanations in the symbolic component of the system.

References

Abraham Bernstein and Esther Kaufmann. 2006. GINO–a guided input natural language ontology editor. In *International Semantic Web Conference*, pages 144–157. Springer.

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.

Enrico Franconi, Paolo Guagliardo, Sergio Tessaris, and Marco Trevisan. 2011. Quelo: an ontology-driven query interface. *Proceedings of the 24th International Workshop on Description Logics*, 745:488–498.

Stephen C. Guy and Rolf Schwitter. 2017. The PENG^{ASP} system: architecture, language and authoring tool. *Language Resources and Evaluation*, 51(1):67–92.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 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.

Henry Kautz. 2020. The Third AI Summer, AAAI Robert S. Engelmore Memorial Lecture. *AAAI'20*. Retrieved November 13, 2021 from https://www.cs.rochester.edu/u/kautz/talks/KautzEngelmoreLectureDirectorsCut.pdf.

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.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777. Curran Associates Inc.

Johannes Rabold, Hannah Deininger, Michael Siebers, and Ute Schmid. 2019. Enriching visual with verbal explanations for relational concepts–combining LIME with Aleph. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 180–192. Springer.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Fabrizio Riguzzi and Damiano Azzolini. 2020. cplint Manual. *SWI-Prolog Version*. Retrieved November 13, 2021 from http://friguzzi.github.io/cplint/_build/latex/cplint.pdf.

Stuart Russell and Peter Norvig. 2020. *Artificial Intelligence: A Modern Approach*. Pearson.

Abdus Salam, Rolf Schwitter, and Mehmet A. Orgun. 2021. HESIP: a Hybrid System for Explaining Subsymbolic Predictions. In *34th Australasian Joint Conference on Artificial Intelligence, Sydney, Australia*. (accepted).

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

Rolf Schwitter, Anna Ljungberg, and David Hood. 2003. Ecole: a look-ahead editor of controlled language. In *EAMT Workshop: Improving MT through other language technology tools: resources and tools for building MT*.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. Retrieved November 13, 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.