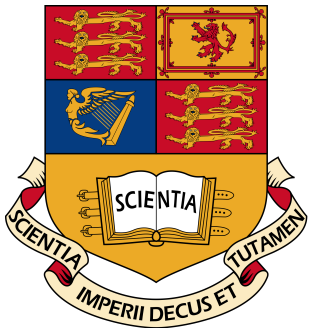


DEPARTMENT OF COMPUTING

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

H-PERL: The Hybrid Property, Event and Relation Learner

Author:
Ross Irwin

Supervisors:
Prof. Alessandra Russo
Dr. Krysia Broda
Dr. Jorge Lobo

June 12, 2020

Chapter 1

Evaluation

This chapter describes the performance details of the components and models outlined in Chapters **??** and **??**. Section 1.1 compares the performance of a selection of counterpart components used in the hardcoded and trained models. Section 1.2 compares the overall performance of the two H-PERL models, and includes details of the models' performance under noisy conditions.

Before presenting the evaluation, we outline the following three key criteria that we want to evaluate the models against:

1. **Accuracy** :- What proportion of questions does the model answer correctly?
2. **Speed** :- How long does the model take to complete the evaluation?
3. **Adaptability** :- How simple is it to transfer the model to a new environment?

While we would have preferred to have compared our models to state-of-the-art neural network implementations for VideoQA, training end-to-end VideoQA networks is very resource intensive and takes a long time. If these models were included in the evaluation, we would add explainability as an additional criterion. However, since our models are both hybrid, their implementations have roughly the same explainability. We do, however, keep in mind that hybrid models are often easier to understand than fully-neural counterparts, as well as being significantly faster to train.

1.1 Components

The first part of the evaluation concerns the individual model components. In this section we compare the performance of the properties, relations and events components from both the hardcoded and trained models. These components are evaluated using perfect input data - where we assume no mistakes have been made in previous parts of the pipeline. We also outline the performance of the

object detection component, which is common to both H-PERL models. All of these components are evaluated using the 200 validation videos, rather than the testing videos, although both sets of videos are generated in the same way.

This section does not include a dedicated discussion on the object tracker's performance, since there are no QA pairs with which it can be directly evaluated. However, after using some intuitive heuristics to gauge the component's object tracking abilities, we see that the tracker assigns identifiers in exactly the way we would expect for the OceanQA environment. In the rest of this chapter, therefore, it should be assumed that the tracker operates with perfect accuracy on the OceanQA environment, but we cannot provide an official evaluation for this.

1.1.1 Object Detector

In Chapter **??** we mentioned that *Intersection over Union* (IoU), which is calculated between a ground truth bounding box and the bounding box produced by the detector, measures how well a detector localises an image. As a reminder, the IoU between an object A and object B is defined as the area of intersection between A and B, divided by the area of union between A and B. Chapter **??** also mentioned that a threshold t can be applied to a set of detections so that only the detections with confidence $> t$ are used. Each IoU threshold that is applied to the set of detections produces a different precision-recall curve for the detector. The average precision (AP) is then defined as the area under the precision-recall curve, although estimates of this value are used. Average precision, however, is only defined for detection of a single class. The mean average precision (mAP) metric is therefore used to find the mean AP across k classes, and is intuitively defined as follows:

$$mAP = \frac{\sum_{i=1}^{k} AP_i}{k} \tag{1.1}$$

In order to conduct the evaluation, we used the *PyCocoTools*[1] library, which implements the metrics used in the COCO object detection challenge. The library provides mAP values for IoU thresholds 0.5 and 0.75, as well as the average mAP over the set of thresholds $\{0.5, 0.55, 0.6, ..., 0.95\}$. Table 1.1 presents mAP values achieved by our detector on the OceanQA validation dataset.

IoU Threshold	mAP
0.5	1.000
0.75	0.998
0.5 : 0.95	0.995

Table 1.1: Mean average precision values for a number of intersection over union thresholds. A threshold of 0.5 : 0.95 indicates an average mAP value for the set of thresholds: $\{0.5, 0.55, ..., 0.95\}$.

[1]Available at: https://github.com/cocodataset/cocoapi

The results presented in Table 1.1 show that our object detector performs very strongly. This isn't particularly surprising, since the dataset environment is very simple and the object detection algorithm used is state-of-the-art. Nonetheless it proves the assumption that it is possible, in the OceanQA environment at least, to extract objects from the frames of the video and work with these objects directly.

Despite the simpicity of the environment, the object detector is, however, quite slow. Detection of all objects in the testing dataset takes about 56 seconds; making the detector the slowest component in the trained model, and the second slowest in the hardcoded model. Figure 1.1 shows the full details of the timings for all components. Since the object detector can be trained for a specific environment, we consider it to be adaptable. Training can take a long time however, and each environment must have a large number of object detection examples available.

1.1.2 Properties

Since the properties component is predicting multiple properties for each object, the component has a separate accuracy for each property. As well as accuracy, we also evaluate the properties component in terms of precision and recall (all three metrics are defined in Chapter **??**), to ensure underrepresented property values are not being missed. Table 1.2 and Table 1.3 show the results for the hardcoded model on the *colour* and *rotation* properties, respectively. We do not present the results for the trained model, since every metric is 1.0 for all property values.

The results show that both the hardcoded and trained models have (almost) perfect performance. It is surprising that the trained model can achieve the same result as the hardcoded model, despite only using weakly-supervised QA training data - data where most objects are not labelled with property values. It is

Colour	Accuracy	Precision	Recall
red	1.000	1.000	1.000
blue	1.000	1.000	1.000
purple	1.000	1.000	1.000
brown	1.000	1.000	1.000
green	1.000	1.000	1.000
silver	1.000	1.000	1.000
white	1.000	1.000	1.000

Table 1.2: Hardcoded properties component results on the colour property.

Rotation	Accuracy	Precision	Recall
upward-facing	1.000	1.000	0.999
right-facing	1.000	0.999	1.000
downward-facing	1.000	0.999	1.000
left-facing	1.000	1.000	1.000

Table 1.3: Hardcoded properties component results on the rotation property.

worth noting, however, that this analysis uses the full-data version of the validation dataset; this means that the object detection was performed without any errors. However, even when the trained object detector was used, Section 1.2 shows that the properties questions were answered almost perfectly.

Despite being made of neural networks, the properties components are both fairly fast. Both components took approximately 18 seconds to label the entire all of the objects in the testing data with properties. As with the detector, both property component implementations are deemed adaptable, since they can simply be retrained for a new environment. One potential issue for the trained properties component, however, is a lack of data. The QA training data is sparsely labelled to start with, and, since training this component requires solving an optimisation problem, there may not be enough data for the optimiser to find a 'good' solution. However, we did not have time to explore this potential issue any further.

1.1.3 Relations

Since the task of the relations component is binary classification, and since the data is highly imbalanced we present the results for each implementation using a confusion matrix. A confusion matrix presents the proportion of true positives (TP), false postives (FP), true negatives (TN) and false negatives (FN) in a matrix. Unlike simply looking at the accuracy, confusion matrices are very useful for ensuring that the model is good at correctly predicting both positive and negative classes. The results for the hardcoded and trained relations components are shown in Table 1.4a and Table 1.4b, respectively.

Table 1.4a shows that the hardcoded relations component achieves an accuracy, precision and recall of 1.0, and therefore an F1-score of 1.0. The trained relations component performs marginally worse, with an accuracy of 0.99, a precision of 0.96, a recall of 1.0, and therefore an F1-score of 0.98. The performance of both components is, however, comparable.

While the hardcoded relations component takes only 22.8 seconds to process the entire testing dataset, the trained component takes almost twice as long, at 39.6 seconds. This time difference is due to the trained component using a neural function, rather than a hardcoded function. However, despite being slower, the trained component is more adaptable than the hardcoded component, since it can learn the definition of binary relations from the data.

Actual: \ Pred:	Yes	No
Yes	0.16	0.00
No	0.00	0.84

(a) Hardcoded relations component.

Actual: \ Pred:	Yes	No
Yes	0.16	0.00
No	0.01	0.83

(b) Trained relations component

Table 1.4: Confusion matrices for hardcoded and trained relations components for the *close* relation. Columns show predicted classifications, while rows show actual.

Event Component	Time Taken (seconds)
Hardcoded	256.8
Hardcoded (EC)	250.5
Trained	5.2

Table 1.5: Time taken in seconds for each event component to detect all events in the testing data. Hardcoded (EC) is the error correcting version of the hardcoded model.

1.1.4 Events

Like properties, detecting events can be considered a multi-class classification task since, for a given pair of frames, the events component needs to select an action from a set, and select an event (or none) from a set. However, in the case of events, both the hardcoded and trained components achieve 100% accuracy. This is not particularly surprising since the rules of the OceanQA environment are relatively simple to model. It does, however, prove that there is no human-introduced bias in the $\mathcal{AL}$ environment model, and that the trained model has learnt correct rules. As with all the previously outlined components, the evaluation of events uses non-noisy data. Section 1.2.2 gives a much better indication of how these components perform under noisy conditions.

As mentioned in Chapter **??**, the speed at which the hardcoded model can operate is a concern. The component evaluation reiterates this; as Table 1.5 shows, the hardcoded events component is about 50 times slower than the trained component on the 200 testing videos.

As well as being very slow, the hardcoded model also comes with very poor adaptability, since the $\mathcal{AL}$ model must be rewritten for every new dataset environment. The trained model is better in this regard, however, the features and feature operations is uses to learn rules may have to be updated for learning new environments.

1.2 Models

In this section we outline the performance of both H-PERL models, as well the error correcting version of the hardcoded model. Section 1.2.1 outlines the performance details of the models on the 200 testing videos, as well as the relative component timings, some of which have already been mentioned above. Then, in Section 1.2.2, we evaluate the models when a faulty detector is used, in order to compare their robustness to changes in the object detection data.

1.2.1 Performance

This section compares the performance of the models, when using the trained object detector, on the 200 testing videos in the dataset. Table 1.6 shows the accuracy - the number of questions answered correctly - as well as the amount of time taken, for each H-PERL model.

Model	Accuracy (%)	Time (seconds)
Hardcoded	97.9	363.3
Hardcoded (EC)	99.1	356.9
Trained	98.8	126.4

Table 1.6: Performance and timing details for each model.

Table 1.6 provides a number of interesting comparisons between the three implementations, some of which are as follows:

1. All three models perform remarkably well; the error correcting hardcoded model, for example, gets only 18 questions wrong, out of a total of 2000. This strong performance is of course partly due to the simplicity of the dataset environment, but it also shows the strength and potential of the H-PERL approach for VideoQA tasks.

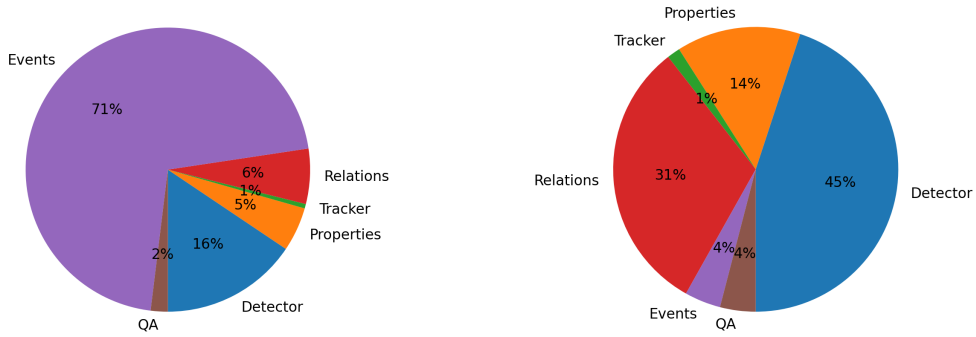
2. The trained model actually performs better than the baseline hardcoded model. An initial analysis shows that the hardcoded model gets a relatively large number of *repetition count* and *repeating action* questions wrong, unlike the trained model (as shown in Table 1.7). We suspect that this may be due to the hardcoded event component's reliance on looking at the entire sequence of frames when detecting a sequence of actions. A small mistake in one frame may lead the component to find a set of actions which differs significantly from the ground truth. The trained event component, on the hand, looks only at two consecutive frames when detecting an action.

3. The error correcting version of the hardcoded model performs better than the baseline version. Despite the performance increase being quite small, this shows that the error correcting algorithm had an unambiguous positive effect on the model's performance. Table 1.7, which provides a breakdown of the models' accuracy on each question type, shows that the performance difference is most pronounced for the questions which relate to events.

4. As alluded to earlier, the trained model is significantly faster than either of hardcoded implementations. This is entirely down to the difference in speed between the hardcoded and trained events components.

Q: / Model:	Property	Relation	Action	CP	RC	RA	ST
Hardcoded	99.7	99.6	97.3	100	98.0	94.6	96.1
Hardcoded (EC)	99.7	99.6	99.0	100	99.2	98.2	98.0
Trained	100	97.5	99.7	99.7	98.4	96.8	99.0

Table 1.7: Accuracy of each model, split by question type. CP refers to *changed-property* questions, RC refers to *repetition count* questions, RA refers to *repeating action* questions and ST refers to *state transition* questions.



(a) Timings for hardcoded model.

(b) Timings for trained model.

Figure 1.1: Breakdown of the relative time taken by each component in both models.

Since speed is a key criteria for our evaluation, we provide the breakdown of the timings for each component for the hardcoded and trained models. The timings for the error correcting version of the hardcoded model are not shown, however, they are almost identical to those of the hardcoded model. These breakdowns are shown in Figure 1.1.

As mentioned previously, the hardcoded model cannot be easily ported to another environment. This is mostly due to its relations and events components, but also because it makes use of the full-data version of the dataset for training its properties component. Other environments will not be able to guarantee that a similar type of dataset is available. The trained mode, on the other hand, is relatively adaptable to other environments, since its components can be re-trained using a different dataset fairly easily.

1.2.2 Robustness