Chapter 1

Hybrid Property, Event and Relation Learner

We outline a novel paradigm for solving the VideoQA problem. This structure merges deep learning and logic-based machine learning and inference methods in an attempt to learn the concepts required to answer the questions. We name this structure: Hybrid Property, Event and Relation Learner (H-PERL). Section 1.1 outlines the structure of an H-PERL model, while Section 1.2 discusses the implementation of a number of H-PERL components which are common to all models shown in this report.

1.1 Architecture

H-PERL is a generic, pipelined structure for finding an answer to a set of questions, given a video. The pipeline is composed of a number of components which, when strung together, form a model (a model can be thought of as a specific 'instance' of the H-PERL pipeline). Chapters **??** and **??** each outline an H-PERL model for the OceanQA dataset.

An H-PERL pipeline assumes that all of the information in an environment which is required to answer the questions can be modeled using: objects; binary relations between objects; and events between two consecutive frames in the video. For many environments, simple environments (like our OceanQA dataset) in particular, this assumption holds. However for many other VideoQA environments, particularly those set in the real-world, this assumption may not be suitable. For example, we may not always be able to extract objects from a video (as in the case of abstract nouns), and yet information on these objects may still be required to answer the questions. Additionally, the H-PERL structure is not capable of modelling relations between objects with an arity larger than 2.

Figure 1.1 shows the components involved in a typical H-PERL pipeline. During evaluation, information from the video and the question flow, from left to

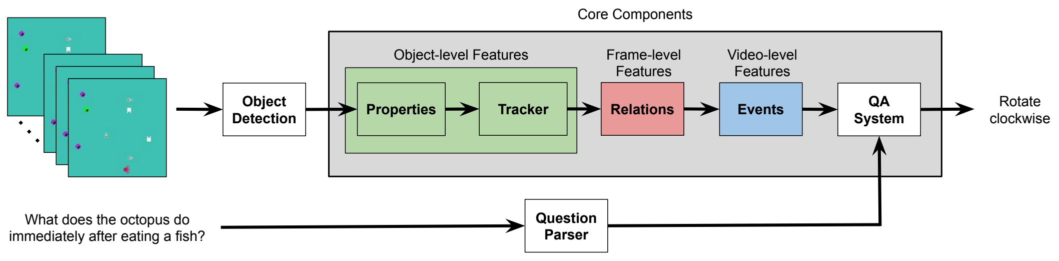

Figure 1.1: An H-PERL pipeline for VideoQA. Green, red and blue shading indicates components which work to extract features at different levels of abstraction. Grey shading indicates the 'core' components of the pipeline.

right, through the pipeline. Each H-PERL model assumes that the object detection and question parsing (TODO add QA system component??) components are "pre-made" (either pre-trained or manually engineered), we refer to these as 'non-core' components. H-PERL allows the remaining, 'core' components to be updated as the model is trained, although they don't necessarily have to be. Each core component in the pipeline accumulates information. This means that each component guarantees to add more features to the data, rather than overwrite existing features (with the small exception of the event component when error correction is used, discussed further in Chapter **??**). As shown in Figure 1.1 components in the pipeline work at different levels of abstraction; the object properties and tracking components work to extract object-level features, while the relations and events components work to extract frame and video-level features, respectively.

The following is a high-level description of the tasks each component is required to complete for the H-PERL pipeline to work with high accuracy:

1. **Question Parser**. The QA parsing component is used to extract relevant pieces of information from the questions (and answers when training). For example, given the question: "What does the octopus do immediately after eating a fish?" and that the question is a state-transition question, the parsing component would extract, firstly, that the object in question was an octopus, and secondly, that the event was 'eat a fish'. The parsing component, therefore, bridges the gap between the symbolic data, which the model works with, and the natural language questions and answers. When an H-PERL model is being evaluated, only the question needs to be parsed. However, when the model is training this component acts as a question-and-answer parser, since the model requires that the feedback that comes from the answer is also in symbolic form.

2. **Object Detector**. The detection component produces bounding boxes and classes for each object in each frame of the video. Any object not detected at this stage of the pipeline is assumed to part of the background and is therefore ignored by the rest of the model.

3. **Property Extractor**. Given a video, the property extraction component assigns a value to every property listed in the environment specification for every object in every frame of the video.
4. **Object Tracker**. The object tracker is required to assign an identifier to each object in a given video. The object identifiers assigned in the initial frame of the video can be arbitrary, but then an algorithm is usually applied inductively to each remaining frame of the video in an attempt to assign each object the same identifier as it was given in the previous frame.
5. **Relation Classifier**.
6. **Event Detector**.
7. **QA System**. Told which type of question is given

TODO assumptions

1.2 Common Components

Give an intro to the each of the components common to all models: object detector, tracker, (QA system), QA parser.