Master's Thesis

FB3 Informatik

To See What No Robot Has Seen Before

Recognizing Objects Based On Natural Language Descriptions

Sehen, was noch kein Roboter zuvor gesehen hat

Objekterkennung auf Basis natürlichsprachlicher Beschreibungen

Author:	Mareike Picklum
Advisors:	Prof. Michael Beetz, Ph.D.
	Dr. Karsten Sohr
Supervisor:	Daniel Nyga, M.Sc.
Submission Date	05.01.2015

Universität Bremen

Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Bremen, January 23, 2019

_____

(signature)

Abstract

This thesis investigates an approach for using natural-language descriptions in the context of object recognition. In particular, we propose a transformation from phrases in natural language to a formalized, computer-understandable representation. We generate a probability distribution of objects and property attributes, in which the attributes are represented by abstract symbols of features. We introduce a number of similarity calculations to measure the relatedness of features and therefore determine the overall similarity between objects. We also design different modelings of a probabilistic relational model and propose a solution for the modeling that suits our object recognition task best. We approve our findings by an evaluation on data sets generated from natural-language descriptions acquired from the internet.

Contents

List of Figures

List of Tables

CHAPTER one

INTRODUCTION

This chapter gives a brief overview of the motivation for this thesis as well as a short introduction to the challenges of using Natural Language (NL) in the context of object recognition. A statement on how this thesis contributes to current research and how it can be used for practical applications is issued.

1.1 MOTIVATION

Executing everyday activities requires quite a number of different skills, of which one of the most substantial ones is the ability to recognize objects. When humans deal with a (whatever natured) instruction, they have to make sure that besides being able to interpret the instruction accurately, they are able to identify all objects necessary to perform the task and, of course, are competent in using them. The same applies to robotics (i.e. household robotics), because to a certain degree of autonomy, a robot is expected to perform a task in a human-like manner. It also needs to be able to interpret an instruction and identify required objects. But what, if one of these prerequisites is not given?

Usually, a robot can recognize things it has seen before. Anything beyond what it has learned is not existent in its world and therefore can not be recognized. In the context of knowledge representation the presumption that only the statements in the evidence are known to be true is called *open world assumption*. Inversely, everything *not* in the evidence is assumed false. In contrast, an open world assumption handles everything but the evidence as unknown.

Analogously, as the robot excludes the existence of anything not seen before its world is called *closed*.

Imagine the following situation, which is illustrated in Figure 1.1: A household

robot is standing in the kitchen of your house and is about to prepare some food (as instructed by you). But from the recipe instructions it requires objects it has never seen or heard of. What will it do? In a *closed* world it can not do anything as it does not have a model to handle the situation properly.

One possible way out could be to equip the robot with capabilities to consult a dictionary and look up a description of the unknown object. Since we live in the 21st century, WIKIPEDIA or some other online source is most likely the reference work of its choice. It could read the article and filter the information in it by what is needed to identify the object. Equipped with this newly acquired knowledge, it can now go on a quest for the unknown object. Once found, comparing the items in its environment to the object description it is provided with, it is finally able to perform the original task: cooking dinner.

This approach is one step further towards an *open* world in which previously unseen objects are recognizable.

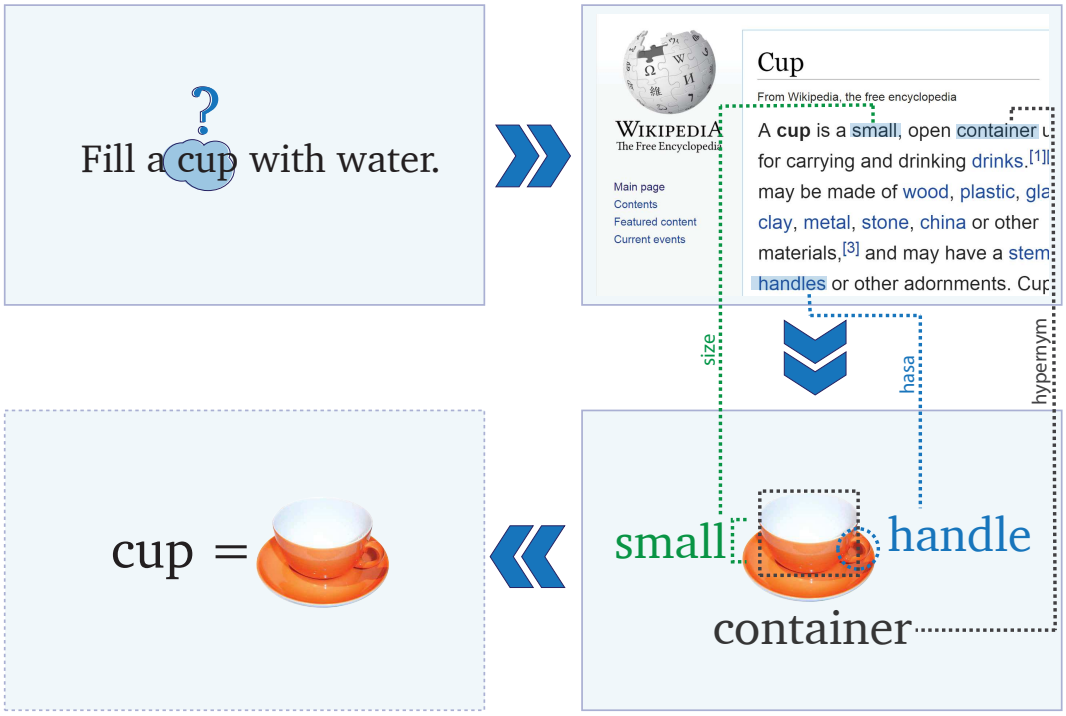

Figure 1.1: Looking up unknown objects: This is an example pipeline, how a human may deal with being confronted with unknown objects.

In the context of autonomous robotics, object recognition has come a long way over the last few years. There exists a variety of different approaches on how to recognize objects, including what kind of features to use and how to interpret them. The typical way to recognize objects is to use a previously trained model, representing feature-object relations in some way. This may be quite complex models like a joint probability distribution or simple lookup-tables matching representing object-feature relations. What these approaches have in common is that the model is typically trained using numerous example instances of the objects that need to

be recognized. The model then learns patterns for objects based on the features occurring in the samples and allows recognizing objects holding the same or similar features. It is therefore necessary to be able to compare features of the perceived object with the ones of previously seen objects represented in a model.

There are different layers of abstraction on the feature level, of which each has its advantages and disadvantages. These layers can be thought of as structured in a hierarchy, as visualized in Figure 1.2, in which each layer builds up upon the respective lower-leveled features.

- **Low-level features** are features that are often closely related to individual instances, that means they are characteristics not necessarily of an object category, but rather of one specific image or object. They may be features such as color cooccurrence- (Chang and Krumm [2]) or gradient histograms, like Scale-invariant feature transform (SIFT) (see Lowe [19]) or Histogram of Oriented Gradients (HOG) (see Dalal and Triggs [4]) or simply values of any color model (Hue Value Saturation (HSV) or RGB). As they have little symbolic meaning and may not be self-explanatory, so humans usually have difficulties with interpreting them. They often occur in form of numerical values which is on the other hand machine-understandable. These values can be compared to the feature values of previously seen objects and therefore allows recognizing objects.

- Using **High-level features** can be seen as the current state-of-the-art approach in object recognition. These features are more abstract than low-level features, and represent a symbolic meaning of the low-level features of an object class. An example for high-level features are color names, as they assign an abstract, human-understandable interpretation to the low-level features (HSV or RGB values). They are intuitively human-understandable and more general than low-level features, as one word (i.e. a color name) is used for a usually larger range of low-level features. Other examples are geometric shapes, which can be generated through corresponding fitting algorithms or transformations (e.g. Hough transformation), or components (part-of relations), which can be determined through image segmentation.

- **Knowledge-based features** add another abstraction layer to the feature level, which can be used to determine similarity relations between them. Abstract symbols in natural language, such as color names have a natural grounding in a taxonomy. This taxonomy links the symbols through relations which can be used to determine their relatedness. For example, the symbols “blue” and “yellow” are not naturally related in any way, but by organizing them in a taxonomy it is possible to determine a connection between them and therefore define a similarity. On the internet one can find a number of taxonomies providing hierarchical structures between objects and ontologies providing information about relations in a certain universe of discourse. The concept of including semantic knowledge in web pages of the world wide web is known as the *semantic web*. By using taxonomy relations to determine a similarity between symbols we create a link to this semantic web.

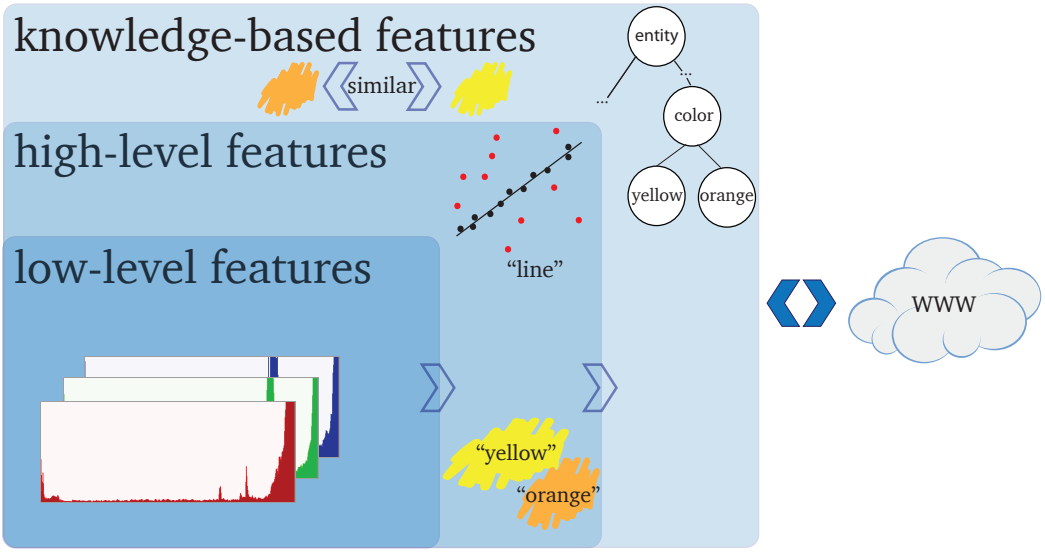

Figure 1.2: Overview over the different feature types. Low-level features are features without any symbolic information such as (color-, gradient-) histograms, SIFT and HOG features. Higher-level features are generated by adding information (e.g. knowledge about geometrics or color values) to low-level features to interpret them in a more intuitive way. Knowledge-based features augment the higher-level features with common knowledge (e.g. similarity information between colors or shapes, or semantic relations between objects) which creates a link to sources, where such common knowledge can be accessed (e.g. the world wide web).

Typically, training objects in an object recognition scenario are examined by means of their local (low- or mid-level) features, but current state-of-the-art research focuses on incorporating high-level features.

1.2 ROBOSHERLOCK

ROBOSHERLOCK[1] is a perception framework for Unstructured Information Management (UIM), which allows to retrieve structured information from unstructured data by applying ensembles of expert systems for annotating information pieces in a multi-stage processing pipeline. A perception system extending ROBOSHERLOCK with probabilistic reasoning about object classification using a trained Markov Logic Network has been proposed by Nyga et al. [24]: To master varieties of properties difficult to perceive, they combine multiple specialized perception methods and organize the perception in a two-step-process. In the first step, multiple annotators add symbolic information regarding their shape, texture, color, size, text pieces, and logos to the perceived scene. To be able to do so, algorithms for clustering (to

[1] `http://www.pr2-looking-at-things.com` (last accessed: January 23, 2019)

decompose the scene into single objects), shape fitting (using Random sample consensus (RANSAC)) and color detection (using RGB or HSV values) are used. The goal is to achieve an abstraction from the feature level in a way that the low-level features are interpreted covering their semantic meaning. After annotating detected objects, a conditional probability $\mathrm{P}(Q|E)$ is used to infer the desired query Q (e.g. the object's identity) from the given evidence E (the symbolic information from the annotators). Employing expert perception methods collaboratively and using the annotation results as evidence for a conditional probability not only allows an increased generality of possible queries against alternative methods, but also eases including more expert systems. The joint probability distribution for the model of scenes, objects and annotations needs to be learned beforehand.

Approaches like this are supposed to approximate a human-like way of representing and recognizing objects which has been found successful on numerous occasions. Using ensembles of classifiers and combining different types of features has shown very successful in different domains and many approaches score remarkable results even on lower quality datasets, which may be affected by illumination or pose/viewpoint variations.

Still the problem of being limited to known classes persists, due to the fact that the resulting models only represent previously seen objects, as all the features are computed from example instances of the respective target classes. The effect of this lack of generality is that classes, of which there are no training instances available, can not be recognized.

It would be desirable to ease this process by allowing to add objects to the model without the need of their physical presence.

1.3 ROBOSHERLOCK WITH KNOWLEDGE-BASED FEATURES

This motivation leads to the idea to use natural language in the context of object recognition tasks and in particular, to train a semantic model from the features that can be found in NL descriptions, where the term "semantic model" in the context of this work refers to a model incorporating knowledge about semantic relations representing perceptual characteristics.

ROBOSHERLOCK currently uses high-level features in the object recognition pipeline. Our approach expands the system to include knowledge-based features by using training data generated from natural-language descriptions.

The question arising from this idea is: Is it possible to recognize objects only from a natural-language description?

This approach adds another abstraction layer on the feature level, as it incorporates knowledge-based features into object recognition tasks and thereby creates a link to the world wide web, in which common knowledge is accessible.

The thought behind this concept is that features in NL context naturally have a grounding in a taxonomy and therefore can be examined by means of their relational characteristics and syntactic dependencies on each other. The information that can be retrieved from such a taxonomy is then augmented with more knowledge about similarities and object relations, signifying one step further towards object recognition using knowledge-based features.

Incorporating the idea of using natural language in object recognition tasks, the following scenario could be thought of, analogous to the situation described above: Given an instruction for a task, an autonomous robot can look up a description in natural language for an unknown object required to perform the task, and either train its internal knowledge base with this information or directly compare it to the objects it has perceived.

The advantages of this approach are, on the one hand, the possibility to generate arbitrary semantic perception models and, on the other, that the necessity to acquire training instances from real objects is not given any more, as the model is trained with natural-language descriptions. To put that in other words, it would be possible to recognize previously unseen objects, which constitutes a significant improvement compared to many other approaches.

Any well-formed NL sentence describing an object can be sifted through to retrieve relevant (i.e. descriptive) information from it. Such an object description could be looked up on one of numerous websites allowing access to semantic networks or dictionaries on the internet.

To name a few examples, ConceptNet[2] is a semantic network representing knowledge by modeling words and their relationships as nodes with labelled edges between them. WORDNET[3] is a Lexical Database using synsets (groups of cognitive synonyms of nouns) to describe concepts. The probably best known source is WIKIPEDIA[4].

WORDNET, in particular, provides rather short and concise descriptions, compared to WIKIPEDIA. A typical description would be *"a small open container usually used for drinking; usually has a handle"* for a *cup* or *"round yellow to orange fruit of any of several citrus trees"* for an *orange*.

Unfortunately natural language is highly ambiguous, for example, there may exist different meanings for one word. As an example, the term "orange" refers to a fruit on the one hand and to a color on the other. We therefore need a method to determine the correct sense of a word, which is another reason why we use WORDNET[5]. It provides the comfortable feature of separating a word into its different senses, which allows us to perform a word sense disambiguation and therefore assign a

[2] `http://conceptnet5.media.mit.edu/` (last accessed: January 23, 2019)

[3] WORDNET can be downloaded on `http://wordnet.princeton.edu/` (last accessed: January 23, 2019)

[4] `http://www.wikipedia.org/` (last accessed: January 23, 2019)

[5] The original WORDNET database has been expanded to include further annotations, which also assign the correct sense to each word in the respective description.

specific meaning to each term.

The information retrieved from these descriptions can be used to train a model (which we use as a knowledge base), with the respective information about objects. This model needs to be represented in a manner that renders a connection between the visual attributes of an object and its identity possible. Particularly interesting attributes are the ones detectable by the robot's respective sensors, such as visual attributes like colors, sizes and shapes. The object attributes detected and annotated by the robot's perception system can then be fed into the model to infer the object's identity.

Like a robot's perception data, NL is generally unstructured, and in order to build up a model, a transformation from the string representation of the NL object description to a formal (robot-understandable) representation is required. From a probabilistic point of view, we are interested in an approach to solving the problem of finding the most probable formal object description, given an informal natural-language specification, that is:

$$\underset{shape,color,size,\ldots}{\operatorname{argmax}} \; \mathrm{P}\left(\begin{array}{c} \text{color(yellow)} \\ \text{color(orange)} \\ \text{hypernym(fruit)} \\ \vdots \end{array} \;\middle|\; \begin{array}{c} \text{“An orange is a yellow or orange fruit.} \\ \text{It is...”} \\ \vdots \end{array} \right)$$

Not only the generation of semantic models from natural-language descriptions is essential about this concept. Even more rewarding is the aspect that by using a joint probability distribution to infer an object's identity, it would even be possible to infer the most significant (=discriminative) attributes needed to describe objects, by querying the distribution of the properties for specific objects or object classes. In particular, we can find the attributes that are most probably used to describe a given object:

$$\underset{shape,color,size,\ldots}{\operatorname{argmax}} \; \mathrm{P}\left(\begin{array}{c} \text{color(yellow)} \\ \text{color(orange)} \\ \text{hypernym(fruit)} \\ \vdots \end{array} \;\middle|\; \text{object(orange)} \right)$$

As an example, a color annotator may be essential for identifying fruit, but not very helpful recognizing other objects, like cups. This information is particularly relevant to identify, which annotators are required to be able to distinguish one object from another.

One advantage of this is the reduction of computation time required for the object inference. When identifying an annotator not needed to recognize a certain object we can do without it and save its complexity.

Another advantage is the reduction of potentially distracting or misleading information. As we concentrate on only the relevant annotations we also make sure that the perception pipeline results in an object description that does not contain too much irrelevant information. If annotations that are not discriminative for the specific object are added the object may be misclassified if its description matches another object.

The transformation from a NL description to a formal representation in the first step and the generation of the knowledge base model in the second, are the core parts of this work and are described in detail in Chapter 2 and Chapter 3.

1.4 RELATED WORK

Elaborate research has been devoted to the field of Natural Language Processing (NLP), including parsing natural language texts (Klein and Manning [13]) and using natural language formulate instructions for robots (Lauriar et al. [17]). Recent research addresses the problem of autonomous robots executing everyday activities instructed in natural language.

Matuszek et al. [21] present how to interpret human language instruction and transform them into formal representation which can then be used for robotic perception and actuation. Particularly, they use natural language to instruct a robot to follow a route through a previously unknown indoor environment. Matuszek et al. [20] present an approach to build a joint model of language and perception for grounded attribute learning. They learn how natural language is represented and extract the meaning of it to ground it in the physical world. They use an online, Expectation Maximization (EM)-like algorithm to train the model to learn language and perception models which is able to identify different types of attributes from novel words. Guadarrama et al. [9] point out that existing approaches expecting queries mapping to pre-trained visual categories fail in retrieving objects described in natural-language phrases and introduce an approach of mapping object images to a set of descriptive words, which are then compared to the natural-language query. The best match represents the object that most likely has been described. Their approach can also handle open-vocabularies to be able to interpret words that have previously not been contained in the training data.

Several research groups are engaged with classification tasks that make use of semantic attributes (Jayaraman et al. [12], Yu et al. [35]), especially in the context of recognizing unseen objects (Su et al. [31]) or activities (Cheng et al. [3]). Using descriptive semantic attributes has shown to be a very useful approach for object recognition tasks.

Duan et al. [5] exercise fine-grained recognition of categories closely related to each other. Primarily, they distinguish bird species by local, discriminative attributes, such as a white belly. They point out that object features should be machine-detectable

and semantically meaningful to humans at the same time, which allows humans to understand the models and can easily contribute domain knowledge. A latent conditional random field model is used to discover discriminative attributes and a recommender system to select semantically meaningful attributes.

Farhadi et al. [6] use descriptions of objects instead of their name to identify one specific instance. In particular, they categorize objects based on their semantic attributes, such as color, material or shape. Their system allows them to report the absence of attributes of familiar objects (such as a missing head of a bird) or, on the other hand, the presence of atypical attributes (like cloth on a motorbike). They use both semantic and discriminative attributes, which they determine using a novel feature selection method, to achieve better classification results. Also they are the first ones to train models on verbal object descriptions instead of training examples, which constitutes a significant advantage over training with large datasets.

Lampert et al. ([16] and [15]) present methods to classify unseen objects by using attribute-based classification, using high-level descriptions of the objects. They introduce two methods to accomplish this: In the first method they train a classifier to be able to infer semantic attributes directly from the object's features. They can then infer the object class by means of class-attribute relations. As these relations are fixed, the inferred class may have been previously unseen. The second approach does not infer the semantic attributes directly, but indirectly. For each training class, multi-class parameters are learned. The posterior distribution of these class labels induces a distribution over the unseen classes through the fixed class-attribute relationship. Thus, after predicting the known classes from the features of the object, its semantic attributes and, subsequently, the unknown class, can be inferred. In doing so, shared information between the object classes are made use of. Similar to this work is the use of higher-level attributes such as colors and shapes. However, the matrix representing class-attribute relations is human-specified and updated manually, instead of using natural-language descriptions to generate a model for this purpose.

1.5 THESIS CONTRIBUTIONS

Many approaches are afflicted with lack of generalization when training models limiting object inference on objects learned beforehand. In this work, we investigate how using natural-language descriptions in object recognition can help solving this issue.

In particular, in our work we address the following:

- We analyze natural-language descriptions regarding their descriptive information in order to infer the object's identity.
- We show that it is useful to learn how to interpret and analyze human language

regarding relevant information and demonstrate how a description knowledge base can be built up hereupon.

- A probability distribution is generated, representing object-attribute relations, which can be used to infer not only the object's identity, but also the most significant and possibly discriminative attributes of it. Such knowledge can be used to create more efficient perception pipelines, as the composition of the ensemble system can be optimized.
- In order to be able to measure the relatedness of objects and features, we present similarity calculations incorporating knowledge about their semantic relations.

The remainder of this document is structured as follows. The concept of this work, whose implementation is explicated in detail in Chapter 3 is outlined in Chapter 2. Section 2.3 briefly introduces Markov Logic Networks and how they can be used to learn the structure of natural-language sentences as well as object-attribute relations.

In Chapter 4 the concept is tested and evaluated.

Chapter 5 contains a summary including findings of this work as well as open questions. Furthermore a perspective for future work is outlined.

CHAPTER two

OBJECT RECOGNITION USING NATURAL LANGUAGE

This chapter first describes the problem of using natural language in the context of object recognition and outlines the basic concept of the approach in Section 2.2. Additional information for understanding methods used in this work is provided, and the core parts of this work is outlined in more detail in Sections 2.4.1, 2.4.2 and 2.4.3.

2.1 USING NATURAL LANGUAGE FOR ROBOTS

Using symbolic attributes is a very intuitive way to describe objects and using them in object recognition tasks enjoys great popularity in recent research. The problem this work addresses is that traditional approaches require numerous training data to train a suitable model.

We therefore propose to generate a perception model trained with data from object descriptions in natural language. These descriptions are acquired from publicly accessible sources on the web and partly incorporate common knowledge.

As natural language is complex, highly ambiguous and manifold in the way that there often exist numerous ways to describe the exact same thing, it is not a robot-understandable information representation.

We propose a method to render the translation from this informal to a formal object description possible. The formal object descriptions can then be used in object recognition tasks, without the requirement of using example instances of the objects for training.

The inference process therefore purely relies on knowledge retrieved from the internet, that is *human* knowledge, instead of only using pre-defined high-level features, thereby creating a link to the world wide web.

2.2 CONCEPT - BASIC

As mentioned in the problem description, we propose using data from natural-language object descriptions to train a perception model. We will now shortly introduce the basics of the underlying concept and which subordinate tasks need to be solved in order to put our concept into practice. Subsequently, the identified core parts are discussed in more detail.

Figure 2.1 visualizes the overall approach in the context of an exemplified object recognition scenario. As shown in the upper box, a perceived object is annotated with a number of symbolic information (features), such as colors, shapes and sizes. The processing pipeline of the visual perception has already been mentioned in the context of ROBOSHERLOCK.

In the perception pipeline of ROBOSHERLOCK a scene is perceived and represented in a 3-dimensional Point cloud which is a set of data points. Using segmentation algorithms, the scene is separated into single objects, called *clusters*, which is why we use this term to refer to a certain object. Note that this is *not* the object's name (or identity), but only a variable identifier. The identifier of a certain object is linked to the actual identity of an object when we determine its name in the inference step.

Different algorithms for clustering, color segmentation, shape fitting or other algorithms, allowing a more abstract interpretation of the low-level features are applied.

The result is a formal representation of the object, based on first-order logic using abstract, symbolic features. The general idea is to find out, which object these features describe by using a model generated from object descriptions from the internet.

The lower box represents the concept implemented in this work. The idea is to train the model using knowledge acquired from the internet to make example instances nonessential to the learning process. We look up suitable object descriptions in online sources like WIKIPEDIA or WORDNET and use the features they contain to generate a probabilistic relational model. The object descriptions we acquire from the web (represented by the cloud in the image) are usually available in natural language, which necessitates a transformation into a formal representation, matching the output of our perception pipeline described above, to achieve comparability.

We generate a model representing relations between the syntactic structure of human language and its information content.

We can use the knowledge that human language has a syntactic structure and follows certain rules, which means that nouns, verbs and adjectives are used in a specific manner, with a certain degree of aberration. This allows making assumptions

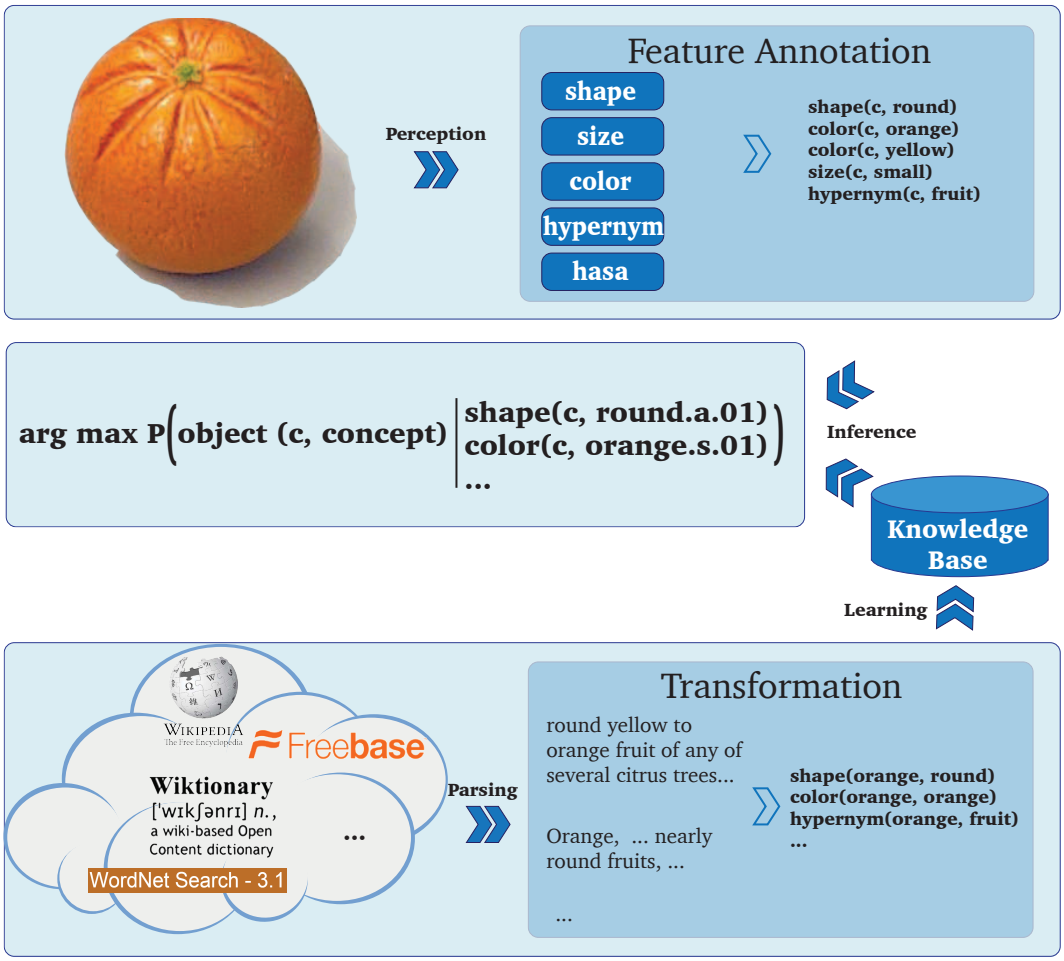

Figure 2.1: Pipeline of the processing steps of the object inference. The upper box shows the perception pipeline in which the perceived object is annotated with a number of symbolic features. The lower box shows that object descriptions are acquired from different sources of the web and then transformed into a formal first-order logic based representation. The generated data is then used to train a probabilistic relational model which is used as a knowledge base. Given the result from the perception pipeline in the upper box, it can be queried for the most probable object identity.

about how words are typically used to describe objects.

We use a parser to transform a sentence in natural language into a syntactic structure describing its grammatical relations. This syntactic structure is represented using predicates in first-order logic. Listing 2.1 shows an example of a parsed sentence.

```
amod(fruit-5, yellow-4)
cop(fruit-5, is-2)
det(fruit-5, a-3)
has_pos(It-1, PRP)
has_pos(a-3, DT)
has_pos(fruit-5, NN)
has_pos(is-2, VBZ)
has_pos(yellow-4, JJ)
nsubj(fruit-5, It-1)
root(ROOT-0, fruit-5)
```

Listing 2.1: Parsing “It is a yellow fruit.”

The predicates are tuples describing typed dependency relations of the form

$$rel \subset p1 \times p2$$

where *rel* is the relation type between *p1* and *p2*. In the example above the relation *amod(fruit-5, yellow-4)* means that *yellow* is the *adjectival modifier* of *fruit*.

A more detailed explanation of the dependency relations relevant in this work is described in Section 2.4.

To train such a model, example descriptions of various objects in natural language are used. In this case, it is not relevant which objects are actually described, as we are primarily interested in the syntactic structure of typical object descriptions. This model therefore incorporates a manual on *how* to extract attribute information (*properties* of objects) from NL descriptions rather than a representation of previously seen objects. This learned model of how object definitions usually look like allows us to make assumptions about the information content of new (unseen) descriptions, so property attributes characterizing an object can be extracted.

The properties as well as the syntactic information are represented formally by predicates in first-order logic, more precisely: A property is defined as a tuple

$$property \subset cluster \times word \times prop$$

where *cluster* is the domain containing identifiers of the described object, *word* contains concepts representing the value of the property type $p_i \in prop$, with $prop = \{COLOR, SIZE, SHAPE, HYPERNYM, HASA\}$. An example would be *property(cluster*$_0$*, yellow.s.01, COLOR)*, which means that object *cluster*$_0$ has the color *yellow*.

In this work, there are five different property types to be mentioned, in particular

- property types representing visual attributes:
  - The **color** may be particularly relevant to describe certain objects, but irrelevant to others (cups, spoons). As an example, fruits are often described using colors, whilst the color is typically not a discriminative fea-

ture for object classes such as vehicles or kitchen utensils, and (if at all) rather used to refer to a specific instance ("My car is red" versus "A car is usually red"). Therefore it can still be used to distinguish certain instances of an object from each other.

- The **size** of an object is usually specified relative to other objects (larger/smaller than, as big as, ...) but may also be absolute (tiny, medium-sized, ...). Still, there is a specification required, indicating how small "small" is and what the thresholds to other sizes are. Absolute size declarations ("a small fruit") can still be used to differ objects from each other.
- A **shape** of an object is often mentioned as a very basic way to describe an object and frequently used in combination with at least one of the other attributes.

- and relational attributes:
  - The **hasa** attribute indicates part-of relations between objects and can be used as a discriminative attribute between similar objects. As an example, cups and bowls are usually described similarly, as they are both small, roundish containers. The fact that cups usually have handles but bowls do not, may be the only difference in their respective descriptions.
  - An object is often described by its superordinate concept and one or more other attributes ("A yellow, oval fruit.", "A small container.", ...). The **hypernym** attribute models the relation between sub- and superordinate concepts.

From a probabilistic point of view, we determine the conditional probability $\mathrm{P}(Q \mid E)$, in which we query about the properties *Q*, and use the syntactic information from the NL sentence as the evidence *E*:

$$\underset{property}{\operatorname{argmax}} \mathrm{P} \left(\begin{array}{c|c} \text{property(cluster}_0\text{, w}_0\text{, SHAPE)} & \text{amod(w}_2\text{, w}_1\text{)} \\ \text{property(cluster}_0\text{, w}_5\text{, SIZE)} & \text{det(w}_0\text{, JJ)} \\ \text{property(cluster}_0\text{, w}_1\text{, COLOR)} & \text{root(ROOT-0, w}_2\text{)} \\ \vdots & \vdots \end{array} \right)$$

This is a simplified formal notation which is specified and explained in more detail in Section 2.4.

With this model we are able to generate formal descriptions of arbitrary objects without the need of any example instances.

Now that we are equipped with a model that renders the translation of an informal object description to a formal one possible, we can use the results of this transformation step as training data to learn object-property relations in a second model. This second model can also be used with ROBOSHERLOCK, which was introduced in Section 1.4, as its formal representation is compatible.

For reasons of reducing complexity, which is explained in Section 2.5 the results of the transformation process described above is converted to a slightly different representation, that is, there is one separate predicate for each type of property. For example, *property(cluster, w_4, SHAPE)* is translated to *shape(cluster, w_4)*, *property(cluster, w_1, COLOR)* to *color(cluster, w_1)* and so on.

This model can be used to reason about the most probable concept (= identity of the object), given the property attributes that have, for example, been perceived by ROBOSHERLOCK.

$$\underset{concept}{\operatorname{argmax}} \mathrm{P}\left(\begin{array}{l|c} \text{object}(\text{cluster}_0, \text{concept}) & \begin{array}{c} \text{shape}(\text{cluster}_0, \text{w}_4) \\ \text{size}(\text{cluster}_0, \text{w}_5) \\ \text{color}(\text{cluster}_0, \text{w}_1) \\ \vdots \end{array} \end{array} \right)$$

We deal with abstract symbols (i.e. the feature values w_x) which can differ in their name but still have the same or similar meaning. For example, a human knows that "round" and "oval" do not differ much from each other, but their formal representations *shape(cluster$_0$, round)* and *shape(cluster$_0$, oval)* is treated as non-identical by a machine as it lacks knowledge about their respective semantic meaning.

We therefore introduce similarity definitions between abstract symbols, based on knowledge we acquire partly from the web (in form of a taxonomy, in which the symbols are arranged) and partly from predefined relations in form of lookup-tables.

In the inference process for an unknown object we therefore further process the evidence from ROBOSHERLOCK and add similarity information to it. In particular, for each feature symbol from the description we add the information about its respective similarity to each of the symbols already incorporated in our trained model. We are therefore able to interpret previously unknown symbols by relating them to known ones.

This is only possible through the link to the semantic web which allows us to determine these similarity relations.

Being able to determine the similarity of two features also allows us to make assumptions about the overall similarity of two object descriptions:

$$\left(\begin{array}{c} \text{shape}(\text{cluster}_0, \text{egg-shaped.s.01}) \\ \text{size}(\text{cluster}_0, \text{small.a.01}) \\ \text{color}(\text{cluster}_0, \text{yellow.s.01}) \\ \vdots \end{array} \right) \overset{sim}{\Longleftrightarrow} \left(\begin{array}{c} \text{shape}(\text{cluster}_0, \text{egg-shaped.s.01}) \\ \text{size}(\text{cluster}_0, \text{small.a.01}) \\ \text{color}(\text{cluster}_0, \text{orange.s.01}) \\ \vdots \end{array} \right)$$

To sum up the subordinate tasks that have to be solved in order to realize our concept, we identify three core parts of the approach, which is described in detail in the

following:

Part I Transforming information comprised in natural-language descriptions into a formal representation based on first-order logic (Section 2.4.1)

Part II Creating a knowledge base from the data generated in Part I to be used for object inference (Section 2.4.2)

Part III Defining a similarity to be able to compare descriptions (Section 2.4.3)

Furthermore we propose using a PRMs for the models that are generated, respectively. A probabilistic relational model (PRM) defines a probability distribution over a set of relational logic interpretations. It incorporates objects with their properties and relations (or the syntactic structure of natural language and its symbolic information content, respectively) and defines a probability distribution over the object attributes given a database containing information about the objects and their respective attributes and relations (Getoor et al. [7]). Having a probability distribution allows us to reason over an arbitrary (but finite) number of predicates. That means that we are not limited to infer an object's identity given some features, but can also reason about frequent attribute combinations or specific attributes that are discriminative for a certain object. Possessing this opportunity, a number of practical applications are imaginable. We can draw conclusions about the algorithms and sensors required to perceive the most informative attributes of an object or design even better models from what we have learned about related attributes.

An introduction to the formal definition of Markov Logic Networks and how they can be used in our work is given in Section 2.3. Implementational details on the account of the introduced models can be found in Chapter 3.

2.3 MARKOV LOGIC NETWORKS

A Markov Logic Network (MLN) is a PRM, specifying a probability distribution over a set of relational logic interpretations. MLNs (introduced by Richardson and Domingos [26] and further investigated by Singla and Domingos [29], [28], [30], [27]) are used to represent knowledge by combining first-order logic (FOL) and probability theory. Formally, an MLN L consists of a set of FOL formulas F and a real-valued weight w_i attached to each of these formulas F_i.

Given a finite set of constants $C = \{c_1, c_2,...,c_{|C|}\}$, an MLN serves as a template for constructing a Markov Random Field (MRF) $M_{L,C} = \langle X, G \rangle$ (depending on the given set of constants, the MRF construction will result in different networks). This MRF has one binary node for each possible predicate grounding in L and one feature for each possible formula grounding in L. More formally, following the definition from Jain [10]:

- *X* is an indexed set of Boolean random variables. For each possible grounding of each predicate appearing in *L*, we add to *X* a Boolean variable (ground atom). We denote by $\mathscr{X} := \mathbb{B}^{|X|}$ the set of possible worlds, i.e. the set of possible assignments of truth values to the variables in *X*
- *G* is an indexed set of weighted ground formulas, i.e. a set of pairs $\langle \widehat{F}_j, \widehat{w}_j \rangle$, where $\widehat{F}_j$ is a ground formula and $\widehat{w}_j$ is a real-valued weight. For each possible grounding $\widehat{F}_j$ of each formula F_i in *L*, we add to *G* the pair $\langle \widehat{F}_j, \widehat{w}_j = w_i \rangle$. With each such pair, we associate a *feature* $\widehat{f}_j : \mathscr{X} \longrightarrow \{0, 1\}$, whose value for $x \in \mathscr{X}$ is 1 if $\widehat{F}_j$ is satisfied in *x* and 0 otherwise, and whose weight is $\widehat{w}_j$.

The probability distribution over the set of possible worlds represented by the MRF is defined as an exponentiated sum of weights of formulas, that are satisfied in *x*:

$$\mathrm{P}(X = x) = \frac{1}{Z} \exp\left(\sum_{j=1}^{|G|} \widehat{w}_j \widehat{f}_j(x) \right) \tag{2.1}$$

which is equal to the exponentiated weighted sum of true groundings of a formula:

$$\mathrm{P}(X = x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{|L|} w_i n_i(x) \right) \tag{2.2}$$

where $n_i(x)$ equals the number of true groundings of F_i in *x* and *Z* is a normalizing constant.

2.3.1 LEARNING AND INFERENCE IN MARKOV LOGIC NETWORKS

An MLN is learned from one or more relational databases under a closed world assumption (i.e. an atom not occurring in the training data is assumed false). The weights of an MLN can be learned using optimization methods (e.g. quasi-Newton or gradient-based). Equation 2.3 shows the derivative of the log-likelihood w.r.t. the weight of the i_{th} formula.

$$\frac{\partial}{\partial w_i} \log P_w(X = x) = n_i(x) - \sum_{x'} P_w(X = x') \cdot n_i(x') \tag{2.3}$$

$P_w(X=x')$ is $P(X=x')$ using the weight vector *w* and the sum is over all possible databases *x'*.

Summing over all possible databases is intractable as well as the computation of the log-likelihood and its partition function *Z* (Richardson and Domingos [26] and Kok [14]) which is required to compute the expectation.

An alternative is therefore an optimization of the pseudo-log-likelihood (Equation 2.4).

$$\mathrm{P}^*_{\mathrm{w}}(X = x) = \prod_{l=1}^{n} \mathrm{P}_{\mathrm{w}}\left(X_l = x_l \,|MB_x(X_l)\right) \tag{2.4}$$

$MB_x(X_l)$ is the state of the Markov blanket of X_l in the data.

Pseudo-log-likelihood can be optimized using algorithms like *BFGS* (Liu and Nocedal [18]).

To penalize large weights which is used for regularization, the likelihood is multiplied by a Gaussian prior *N(0, σ)* on each weight which is calculated as follows:

$$prior' = prior - \frac{1}{2 \cdot \sigma^2 \cdot w_i^2} \tag{2.5}$$

where σ denotes the standard deviation of the Normal distribution.

As an MLN represents a joint probability distribution we can answer arbitrary queries about the objects incorporated in the model ([26]):

$$\begin{aligned} \mathrm{P}\left(F1 \,|F2, L, C\right) &= \mathrm{P}\left(F1 \,|F2, M_{L,C}\right) \\ &= \frac{\mathrm{P}\left(F1 \wedge F2 \,|M_{L,C}\right)}{\mathrm{P}\left(F2 \,|M_{L,C}\right)} \\ &= \frac{\sum_{x \in \mathcal{X}_{F1} \bigcap \mathcal{X}_{F2}} \mathrm{P}\left(X = x \,|M_{L,C}\right)}{\sum_{x \in \mathcal{X}_{F2}} \mathrm{P}\left(X = x \,|M_{L,C}\right)} \end{aligned} \tag{2.6}$$

Here, *F1* and *F2* are formulas in first-order logic, *C* is a finite set of constants, *L* is an MLN and $\mathcal{X}_{F_i}$ is the set of worlds where F_i holds.

The computation of Equation 2.6 directly is intractable since inference in MLNs subsumes probabilistic and logical inference, which are #P-complete and NP-complete, respectively.

To be able to use solving methods already available the Maximum a Posteriori (MAP) inference problem in an MLN can be converted into a weighted constraint satisfaction problem (WCSP).

Following the definition from Jain et al. [11], a WCSP is defined as follows: A weighted constraint satisfaction problem (WCSP) is a tuple $\mathcal{R} = \langle$ *Y, D, C* $\rangle$:

- $Y = \{Y_1, ..., Y_n\}$ is a set of *n* variables.
- $D = \{D_1, ..., D_n\}$ is the collection of the domains of the variables in *Y*, such that $D_i = dom(Y_i)$ is the domain of Y_i. For a given variable Y_i we may also denote its domain by D_{Y_i}. We write D_S to denote the Cartesian product $\prod_{Y_i \in S} D_i$ for some subset of the variables $S \subseteq Y$. $\mathcal{Y} := D_Y$ denotes the Cartesian product of all domains and hence represents the set of possible variable assignments.

- $C = \{c_1, ..., c_r\}$ is a finite set of r soft constraints. A soft constraint c_i is a function on a sequence of variables V from the set Y (V is called the *scope* of the constraint) such that c_i maps assignments (of values to the variables in V) to cost values: $c_i : D_V \rightarrow \{0, ..., \top\}$. If an assignment is mapped to $\top$, it is considered inconsistent.

- A solution to $\mathcal{R}$ is a consistent assignment to all variables. An optimal solution $y \in \mathcal{Y}$ minimizes the accumulated cost $\sum_{i=1}^{r} c_i(y)$ over all constraints (we assume that y is implicitly projected to the actual scope of c_i).

This transformation is possible because each formula in an MLN can be viewed at as a generalized soft constraint.

More detailed, each formula in the MLN is transformed in a way that they have positive weights. This is carried out by negating all formulas along with their weights (if the weight was negative before), without changing the semantics of the MLN (Jain et al. [11]).

We therefore transform each formula-weight pair (F_k, w_k) to $(\neg F_k, \neg w_k)$. A positive weight of a formula increases the probability of any possible world satisfying it and inversely, can be interpreted as a cost value in case the formula is *not* satisfied.

Therefore the definition of c_i looks as follows:

$$\begin{aligned} c_i(s(x)) &= \begin{cases} \hat{w}_i & \text{if } \hat{f}_i(x) = 0 \\ 0 & \text{if } \hat{f}_i(x) = 1 \end{cases} \\ &= (1 - \hat{f}_i(x)) \cdot \hat{w}_i \end{aligned} \tag{2.7}$$

$\hat{f}_i$ is the feature function associated with $\hat{F}_i$ and $s(x)$ is implicitly projected to the actual scope of c_i.

The costs are then scaled because each cost c in a WCSP needs to be a natural number, while the weights from the MLN are in $\mathbb{R}_0^+$.

A solver for weighted contraint satisfaction problems can now be used to estimate an efficient most probable explanation (MPE), that is the most likely assignment for any instance of the MLN.

2.3.2 ADVANTAGES OF MARKOV LOGIC NETWORKS

In general, an MLN maintains a joint probability distribution over observations from whole databases. This makes it possible to answer arbitrary queries about the trained model which allows reasoning about different features of objects, for example.

Another advantage of using MLNs is that errors or contradictions in the training databases do not result in an entirely corrupt model, as they can handle such uncertainty by learning systematic errors in the dataset and treating them in a meaningful way. The convenient representation in first-order logic facilitates the extension of the model, as new predicates can simply be added and incorporated into the existing template formulas.

The downside, however, is that learning as well as inference can be computationally very expensive, which needs to be taken into account during modeling the MLN.

In this work, one MLN uses syntactic evidence from NL descriptions to reason about object properties, and a second one uses these properties as evidence to reason about the object's identity. A more detailed description of the process pipeline is presented in the next chapter.

2.3.3 EXAMPLE OF A MARKOV LOGIC NETWORK

A very simple example of an MLN may look like the following: We assume we have defined three different predicates

$$object \subset obj,$$

$$shape \subset shp$$

and

$$color \subset col$$

or in FOL notation:

```
object(obj)
shape(shp)
color(col)
```

Listing 2.2: MLN predicates

Each of the predicates can take one argument, whose values are from different domains than the respective others. In that way we use a *typed logic*. In the grounding process an *object* atom can only take a value from the *obj* domain as its argument, a *shape* atom from the *shp* domain, and so on.

Furthermore we design one single formula, assuming a relation between objects and the attributes shape and color.

```
0 object(+?o) ^ shape(+?s) ^ color(+?c)
```

Listing 2.3: Example formula

The formula is constructed using *PRACGrammar*, which is a slightly modified version of the standard syntax for MLNs. 0 is assigned as an initial weight. Variables

are prefixed with a "?", while everything else is considered a constant. As some variables are prefixed with a "+", the formula is actually a formula template, from which another formula for each possible grounding of the variables is generated, that means, for each binding of the variable to a value in the respective domain.

In our small example, we assume the following settings of the domains: There only exist two different objects (apples and tomatoes), two different colors (red and green) and only one shape (round) or more formally: *obj* = {*apple, tomato*}, *col* = {*red, green*} and *shp* = {*round*}.

According to the definition above, the formula template in Listing 2.3 is grounded to the formulas in Listing 2.4 as each formula containing a "+"-prefixed variable is grounded to another formula for each variable assignment from the respective domain.

```
0 object(apple) ^ shape(round) ^ color(red)
0 object(apple) ^ shape(round) ^ color(green)
0 object(tomato) ^ shape(round) ^ color(red)
0 object(tomato) ^ shape(round) ^ color(green)
```

Listing 2.4: Example formula

Depending on the training databases, the formula weight is updated according to the number of training examples, that evaluate the respective formula true. The weights have to be viewed relative to the other formula weights. Only knowing that one formula has a specific weight does therefore not help to make an assumption when querying an object's identity, for example.

The resulting ground MLN maintains a joint probability distribution over objects, shapes and colors, and we can query an arbitrary variable or variable combination.

If we assume, according to our training data we have calculated (or simply set) the weights as follows:

```
2 object(apple) ^ shape(round) ^ color(red)
4 object(apple) ^ shape(round) ^ color(green)
3 object(tomato) ^ shape(round) ^ color(red)
0.5 object(tomato) ^ shape(round) ^ color(green)
```

Listing 2.5: Example formula

A red and round object may be a tomato as well as an apple, the same applies to green and round objects. So each of the above combinations is *possible*, but not equally *probable*.

This small example already shows the complexity of MLNs and it Looking at a manageable situation like this small example we can easily observe the relations between the objects and their attributes. Having these relations in mind, one can also accept the tendency of the MLN to classify green, round objects to be an apple rather than a tomato.

From Section 2.3.1 we understand that given larger domains and more complex formulas, the resulting ground MLN may be incomprehensible due to its complex

structure.

2.4 CONCEPT - DETAIL

Having introduced the formal definition and application purpose of MLNs, we will now discuss the core parts of the concept in detail.

2.4.1 TRANSFORMING NATURAL LANGUAGE

Natural language is used whenever humans want to communicate with each other. We use it in our everyday life to interact with others, to formulate questions, instructions or descriptions of things, or in other words, we use it to deliver information. When asking someone to pass a specific object, we use language with rich semantics to describe it as accurate as possible, which includes naming basic-level or fine-grained categories. In doing so, we assure the person we are addressing understands which object we desire and is able to distinguish it from other, possibly similar ones.

We are even able to specify one explicit instance of an object class by adding more detailed information like brand-names or instance-specific attributes. A robot will not understand object descriptions in natural language, but rather needs a formal representation to be able to interpret them. Using NL descriptions to train a model provides the possibility to use descriptions retrieved from the internet, for example WIKIPEDIA or WORDNET. Tenorth et al. [32] investigate natural-language instructions and present methods how to transform them into a formal, logic-based representation.

Similarly, in this work, natural-language descriptions are parsed and processed to retrieve the contained information from it. We already pointed out the advantages of using natural language and identified a number of difficulties we are facing.

The Stanford parser[1] (Klein and Manning [13]) is a probabilistic context-free grammars (PCFG) parser, which retrieves the syntactic structure of natural language, such as part-of-speech (POS) tags, adjectival or adverbial modifiers and other grammatical properties. This information makes it possible to find patterns in the syntactic structure of a phrase and its related information content.

If the grammatical structure of the (query) sentence is incorrect the resulting output of the Stanford parser will not be as accurate as if it follows grammar rules. This is

[1] Stanford Parser: `http://nlp.stanford.edu/software/lex-parser.shtml` (last accessed: January 23, 2019)

due to the fact that the parser is newswire-trained[2] which means that the training data is taken from newspapers. Newspapers are using complete sentences with usually correct grammatical structure and the trained model incorporates this way to use the language. Complete sentences as can be found in WIKIPEDIA descriptions are therefore not a problem.

In contrast, object descriptions in WORDNET are usually short and often consist of a number of clauses, which do not necessarily need to be full, grammatically complete sentences. The difference in the results from the Stanford parser given the clause "a round, yellow fruit" and the complete sentence "It is a round, yellow fruit." can be seen in Listings 2.6 and 2.7. While in the first parsing result, "round" is considered a noun (*has_pos(round-2,NN)*), it is correctly determined to be an adjective (*has_pos(round-4,JJ)*) in the second. This is important because the word sense disambiguation following next is based on the POS tags, the Stanford parser returns.

```
amod(fruit-5, yellow-4)
appos(round-2, fruit-5)
det(round-2, a-1)
has_pos(a-1, DT)
has_pos(fruit-5, NN)
has_pos(round-2, NN)
has_pos(yellow-4, JJ)
root(ROOT-0, round-2)
```

Listing 2.6: Parsing "a round, yellow fruit"

```
amod(fruit-7, round-4)
amod(fruit-7, yellow-6)
cop(fruit-7, is-2)
det(fruit-7, a-3)
has_pos(It-1, PRP)
has_pos(a-3, DT)
has_pos(fruit-7, NN)
has_pos(is-2, VBZ)
has_pos(round-4, JJ)
has_pos(yellow-6, JJ)
nsubj(fruit-7, It-1)
root(ROOT-0, fruit-7)
```

Listing 2.7: Parsing "It is a round, yellow fruit."

Each syntactic information returned by the Stanford parser represents a dependency in form of binary relations. Such a relation holds between a so-called governor (or head) and a dependent. Particularly interesting in our case are adjectival modifiers (*amod*), which give important information about an adjectival phrase modifying the meaning of a noun. The expression "A yellow fruit." would therefore reveal the dependency *amod(fruit, yellow)*. Also, nominal subjects (*nsubj*, the syntactic subject of a clause), direct objects (*dobj*) and objects of preposition (*pobj*, the head of a noun phrase following the preposition) represent word relations in the descriptions, from which the described object properties can be inferred.

Some dependencies are - in our case - only useful in correlation with the POS tags of the respective components as we are primarily interested in adjectives (colors, sizes, shapes..) and second-rank, nouns (hypernyms, parts). An example for such a dependency is *nsubj* which is used to reveal the nominal subject of a clause as the

[2]More information about the training sets used for the Stanford parser can be found on `http://nlp.stanford.edu/software/parser-faq.shtml#z` (last accessed: January 23, 2019)

```
amod(fruit-5, yellow-4)
cop(fruit-5, is-2)
det(fruit-5, a-3)
has_pos(It-1, PRP)
has_pos(a-3, DT)
has_pos(fruit-5, NN)
has_pos(is-2, VBZ)
has_pos(yellow-4, JJ)
nsubj(fruit-5, It-1)
root(ROOT-0, fruit-5)
```

Listing 2.8: Example for full syntactic evidence: “It is a yellow fruit.”

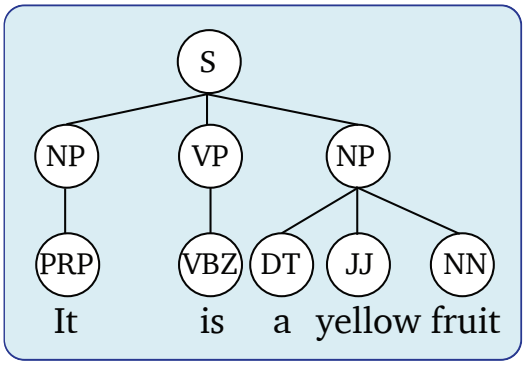

Figure 2.2: Example parse tree for “It is a yellow fruit.”

dependent. A special case we take advantage of is the occurrence of copulas. A copula is used to link a subject with a predicate (its *complement*). If such a copular verb occurs in the sentence the *nsubj* dependency will not reveal the verb itself as the *head* but its *complement*. This can be either an adjective or a noun.

Parsing “The lemon is yellow” returns (amongst others) *nsubj(yellow, lemon)*. Similarly, a dependency (*cop*) for such a copula, which relates copular verb to its complement, is another example. This is especially interesting for retrieving hypernyms, as “A lemon is a yellow, oval fruit.” gives us *cop(fruit, is)* as a result.

In linguistics, a copula (plural: copulas or copulae) is a word used to link the subject of a sentence with a predicate (a subject complement), such as the word is in the sentence "The sky is blue." The word copula derives from the Latin noun for a "link" or "tie" that connects two different things.

The full syntactic evidence from the sentence “It is a yellow fruit.” can be found in Listing 2.8, Figure 2.2 is a tree representation of the parsing result.

This result is then fed into an MLN to make it possible to identify *yellow* and *oval* as ascribing adjectives of type *color* and *shape*, respectively and *fruit* as a potential hypernym. Note, that the added numbers denote the position of the respective word in the text to clarify, which word is referred to, if one term occurs multiple times.

Figure 2.3 illustrates the steps for the transformation process. After parsing a natural-language description (here: “It is an orange or yellow fruit.”), the resulting syntactic information is supplemented by further annotations, which will be explained in detail in the following.

Property tags are added manually providing information about the object's attributes contained in the descriptions. In this case, two attribute tags for color (yellow and orange), and one for hypernym (fruit) would be added to the training example.

The property types are mostly attributes that can be perceived visually, as we concentrate on a perception system for household robots. The *hasa* relation somehow belongs to both categories (visual and relational attributes), as it may indicate a visible or non-visible part of the object. A cup has a handle, which is perceivable,

if the cup is viewed from the right angle and the perception pipeline provides an annotator capable of detecting handles. On the other hand, a cherry has a stone, which is usually not visible as it is inside the fruit.

Other attributes (material, consistency, ...) may be included, depending on the contingency of sensors and perception algorithms available. Here, we concentrate on the five features mentioned above.

We already introduced the idea of using similarity definitions for the abstract symbols, to be able to compensate variations in natural-language descriptions.

These similarities are incorporated in our MLN by generating new atoms. For each symbolic feature in the evidence we calculate its respective similarity to the other symbols already incorporated in the model. Until now the evidence database only contains atoms for the features retrieved in the perception pipeline. These are concidered certain. Everything that has *not* been perceived is considered false. So far we observe a *binary* condition of the truth of the evidence. It is either not existent and therefore considered false or it is present in the evidence database and therefore considered true. For the similarity incorporation we need to adapt to the fact that we have multi-valued truth values for our evidence features. For example, if we perceive the color *blue* for an object we add to the evidence database *color(blue)* with a certainty of *1* but also *color(yellow)* if the color *yellow* is already incorporated in the MLN. The truth value for the new atom is neither *0* nor *1* but the value representing the similarity between *blue* and *yellow*.

The calculus of the logical rules in the MLN therefore needs to be adapted accordingly which means that we make a step from strictly binary first-order logic to a multi-valued logical calculus. In order to do so, we use Fuzzy Logic which is a multi-valued extension of first-order logic and is based on the theory of fuzzy sets Nyga and Beetz [23].

Allowing real-valued degree of truth for the evidence predicates makes it possible to perform learning and inference tasks in the presence of vague evidence.

We have mentioned that the second argument of the *property* predicate (e.g. w_0 in *property(cluster$_0$, w_0, SHAPE)*) is of the abstract type *word* and contains concepts representing the value of the respective property type (the third argument). To specify this further, these concepts are represented by WORDNET synsets, which allows us to assign a certain meaning to the possibly ambiguous abstract symbol.

The training set for the MLN thus contains not only the syntactic evidence provided by the Stanford Parser and property information, but also the necessary information about word senses. In particular, next to its POS tag and syntactic relation to other words, each term is assigned a WORDNET sense, to disambiguate its meaning.

For new natural-language descriptions, we therefore not only reason about the most probable properties given the syntactic evidence, but we also infer the most probable word senses (*has_sense*) of all property values, given additional information about their similarity to known concepts from the model. This additional information is added by retrieving all possible WORDNET synsets for each word in the natural language sentence and calculating its similarity to concepts already incorporated in

the model.

We acquire this similarity by incorporating a taxonomy in which concepts are structured using different types of relations provided by WORDNET. By interpreting these relations, we can define similarities between words, which leads to the straightforward idea that using this similarity can help to recognize objects by determining a similarity between the features from two object descriptions.

The definition of the similarity of two concepts are introduced in Subsection 2.4.3.

We query the properties and word senses (*has_sense*), and use the syntactic information from the parsed NL sentence as well as word similarities between known concepts and possible word senses (*is_a*) as the evidence, i.e.: The (updated) conditional probability of the property extraction therefore looks like the following:

$$\underset{property,has\_sense}{\operatorname{argmax}} \; \mathrm{P}\left(\begin{array}{c|c} \mathrm{property}(\mathrm{cluster}_0, \mathrm{w}_0, \mathrm{SHAPE}) & \mathrm{amod}(\mathrm{w}_2, \mathrm{w}_1) \\ \mathrm{property}(\mathrm{cluster}_0, \mathrm{w}_5, \mathrm{SIZE}) & \mathrm{det}(\mathrm{w}_0, \mathrm{JJ}) \\ \mathrm{property}(\mathrm{cluster}_0, \mathrm{w}_1, \mathrm{COLOR}) & \mathrm{root}(\mathrm{ROOT\text{-}0}, \mathrm{w}_2) \\ \mathrm{has\_sense}(\mathrm{w}_0, \mathrm{s}_0) & \mathrm{is\_a}(\mathrm{s}_0, \mathrm{concept}\_x) \\ \vdots & \vdots \end{array} \right)$$

Note, that using a PRM allows us to reason about multiple variables, as the model represents a probability distribution over the syntactic properties and their relations to the properties and word sense references.

We proposed to use MLNs for the model implementation, which is generated by designing formula templates, which will be extended to multiple formulas in the grounding process (see Section 2.3).

An excerpt of the formula templates for the MLN used for inferring the property attributes is shown in Listing 2.9, the complete formula set can be found in the appendix (see MLN Templates - Property attributes).

```
1 0 nsubj(?w1, ?w2) ^ has_pos(?w1, JJ) ^ has_sense(?w1, ?s1)
      ^ property(?w1, +?prop) ^ ?w1 =/= ?w2
2 ...
3 0 amod(?w1,?w2) ^ has_pos(?w2, JJ) ^ has_sense(?w2, ?s2) ^
      is_a(?s2, +?c) ^ property(?w2, +?prop) ^ ?w1 =/= ?w2
4 ...
5 0 cop(?w1,?w2) ^ has_pos(?w1, NN) ^ has_sense(?w1,?s1) ^
     is_a(?s1, +?c) ^ property(?w1, HYPERNYM) ^ ?w1 =/= ?w2
6 0 prep_with(?w1,?w2) ^ has_sense(?w2,?s2) ^ property(?w2,
     HASA) ^ ?w1 =/= ?w2
7 ...
```

Listing 2.9: Excerpt of MLN used for property extraction

The atoms *amod*, *nsubj*, *has_pos* and *cop* in the above formulas represent syntactic

information in the natural language sentence, the atoms *has_sense* and *property* are queried.

From the formulas you can see how the syntactic evidence described previously is interpreted in relation to their semantic meaning. Each of the formula templates is a conjunction of atoms with an initial weight of *0* and can be understood that way, that each of the prerequisites (= the atoms) must be true to create an overall true statement.

The first formula template would be grounded to one new formula for each possible property type, as the variable *prop* is prefixed with a "+".

The weight of the formulas is determined, depending on for how many of the training examples this formula is a true statement, i.e. how many words are adjectives (i.e. have *JJ* as a POS tag), are a first argument of a *nsubj* relation and have been assigned the respective property type.

This formula's purpose is to determine adjectives in formulations like "The lemon is yellow", in which a copular verb ("is") occurs. In those cases, the *nsubj* does not return the verb itself as the governor of the relation, but its complement ("yellow"). By restricting the governor (variable *w1*) to be an adjective (*has_pos(?w1, JJ)*), we further constrain the results to only contain the desired information.

The second formula directly addresses adjectival modifiers, which is a bit more straight-forward.

The formula in line 5 can be interpreted similarly to the first one, it is used to extract those adjectives, that are the complement of a copular word. The difference is that we are interested in nouns this time (*has_pos(?w1, NN)*), because we try to infer the hypernym (*property(?w1, HYPERNYM)*) instead of an arbitrary property.

The last formula template in this excerpt makes use of the convenient *prep_with* predicate, which is returned whenever the term "with" occurs in a sentence. As the word "with" is often used to describe part-of relations ("a container with a handle"), we take this as a strong evidence for variable *w2* being a property of type "HASA".

The remaining formula templates are each specialized on different syntax-property relations, but their structure is the same as in the examples above.

Now we are able to transform arbitrary natural-language descriptions into a formal one and can generate a model representing object-attribute relations, which can be used as a knowledge base to infer an object's identity in the next section.

2.4.2 OBJECT INFERENCE

From Section 2.4.1 we can see, that we are now able to generate formal, first-order logic based descriptions for arbitrary objects by transforming their respective natural-language descriptions acquired from the internet.

These formal descriptions can now be used to train another MLN, which can then

be used as a knowledge base for object inference tasks.

By designing formulas, that are conjunctions of predicates representing object identities and different types of features, we learn a probability distribution over the respective object-attribute relations.

In this case, we design the following formulas explained below.

```
1 0 object(?c, +?objID) ^ size(?c, +?shape) ^ shape(?c, +?
    col)
2 0 object(?c, +?objID) ^ color(?c, +?size) ^ shape(?c, +?
    col)
3 0 object(?c, +?objID) ^ color(?c, +?size) ^ size(?c, +?
    shape)
4 0 object(?c, +?objID) ^ color(?c, +?shape) ^ hypernym(?c,
    +?col)
5 0 object(?c, +?objID) ^ shape(?c, +?shape) ^ hypernym(?c,
    +?col)
6 0 object(?c, +?objID) ^ size(?c, +?size) ^ hypernym(?c, +?
    col)
7 0 object(?c, +?objID) ^ hasa(?c, +?part) ^ hypernym(?c, +?
    col)
8 0 object(?c, +?objID) ^ hasa(?c, +?part)
9 0 object(?c, +?objID) ^ hypernym(?c, +?hyp)
```

Listing 2.10: MLN used for object inference

The above formulas represent common combinations of attribute pairs used to describe an object. These combinations follow the observation that when describing an object in natural language, it is common to not use more than two adjectives. In particular one tends to name a superordinate concept of the given object, together with one more (usually visual) attributes, to describe it in one sentence.

Each formula can therefore be understood in the way that "the object x is likely to be described using attribute a and b". When designing formula templates for object-attribute relations, it is helpful to have an idea of the combinations that usually occur. Creating too short formula templates (e.g. all formulas are formed like *object(?c, +?objID)* $\hat{}$ *rel(?c, +?part)*, *rel* $\in$ {*color, size, shape, hasa, hypernym*}), may result in a model that is too general and not very expressive. On the other hand, if the formula templates are too long, the model will be too specific and may result in non-satisfiable formulas.

In the worst case, if a domain of a certain predicate is empty, no formula containing this predicate can be grounded as no valid variable assignment is possible.

The configuration in Listing 2.10 has proven successful, as it not only contains all combination pairs of the visual attributes *color, shape* and *size* in addition to separate formulas for *hasa* and *hypernym*, but also additional formula templates for all combinations of *hypernym* and a second attribute type. The two templates in line 8 and 9 are the only ones with only two atoms.

Once the MLN is trained using the output generated in Part I, we may use the model

to recognize objects. A perception system generating an object description using the predicates we introduced, may be ROBOSHERLOCK.

We augment the data with our previously introduced similarity information, to be able to compare features, without requiring their abstract symbols to be identical.

In particular, we add an extra atom for each feature occurring in the object description and each feature incorporated in the model, under consideration of its respective type. That means that for an evident feature *type(c_x, a)*, we add another feature *type(c_x, b)* for each $b \in t_d$, t_d being the domain a is in. As an example, if our evidence only contains *color(c, yellow.s.01)* and the domain *color* contains the concepts (or, as we know WORDNET synsets) *yellow.s.01* and *blue.s.01*, the evidence database looks like the following afterwards:

```
x color(c, blue.s.01)
1 color(c, yellow.s.01)
y color(c, orange.s.01)
```

Listing 2.11: Example for updated evidence

x and *y* are the values for the similarity calculated for *yellow.s.01* and the elements from the *color* domain, respectively.

Therefore, the presence of an attribute is not a binary statement anymore (either a certain color has been perceived or not), but incorporates a fuzzy semantics. In this case, we have not perceived the color *orange.s.01*, but something similar to it. This makes it possible, to infer objects from a description, that does not entirely match the relations represented in the model. If a lemon is related to the color *yellow.s.01*, it is now possible with the updated evidence, to assume it might describe a lemon with a certain probability.

If this assumption is useful or not, strongly depends on the similarity measure we use. If the similarity indicates a high similarity between semantically unrelated features, the inference will output a result, which is completely unrelated to the actual description.

In the following, we propose a similarity calculation that has shown useful as it reflects the intuitive understanding of the relatedness of different features.

2.4.3 SIMILARITY

We mentioned the use of a similarity calculation to support our proposal of using natural-language instructions for object recognition. We motivate the use of this similarity calculation, explain how it benefits our approach and define in detail, how the similarity reflects the relatedness between concepts.

Similarity Usage

The implementation of the modules comprises a similarity calculation to infer the most probable WORDNET word senses for the terms used in the object description and to determine the overall similarity of the described object to an object in the knowledge base.

The similarity calculation introduced here makes it possible to compare words based on their symbolic meaning. In our implementation, the similarity calculation has two main functions:

- Render the word sense disambiguation in Part I possible
- Allow an intuitive (human-like) comparison of single features and subsequently, whole object descriptions

In the context of the word sense disambiguation, the similarity serves the purpose to indicate, if related words have been used before (e.g. occur in the training set). Before the actual inference is started, the database containing the evidence from the Stanford parser, that is, the syntactic evidence from the input description, is extended by similarity information for each word.

To do that, every possible word sense for a term is retrieved from WORDNET, of which each is compared to words already incorporated in the MLN.

As an example, the word "yellow" has 8 different meanings according to WORDNET (see Table 2.1).

Word	Synset	Description
yellow	jaundiced.s.01	affected by jaundice which causes yellowing of skin etc
	yellow.n.01	yellow color or pigment; the chromatic color resembling the hue of sunflowers or ripe lemons
	yellow.s.01	of the color intermediate between green and orange in the color spectrum; of something resembling the color of an egg yolk
	yellow.v.01	turn yellow
	chicken.s.01	easily frightened
	yellow.s.03	changed to a yellowish color by age
	scandalmongering.s.01	typical of tabloids
	yellow.s.05	cowardly or treacherous

Table 2.1: Word senses: the eight different meanings of the word "yellow" according to WORDNET with their respective descriptions.

As a human, we intuitively know from the context, how to interpret the word. In the sentence "A yellow, oval fruit.", we know that *yellow* refers to a color. On the other hand *yellow* in the sentence "He is too yellow to stand a confrontation" means being cowardly. We take certain knowledge into account, for example, the part of speech of the word and the context it is used in.

A robot (or machine, in general) is not able to do so, unless we teach it. We therefore narrow down the number of possible word senses in the first place, by only allowing the ones of the same part of speech as the word is in. We can use the POS tag from the syntactic evidence on this account.

By adding similarity information for all remaining word senses with each concept in the MLN (which can be seen as the context in our case), we have a good chance of picking the "correct" word sense in this particular sentence.

Sticking with our example sentence "A yellow, oval fruit.", we would ignore the meanings *yellow.n.01* and *yellow.v.01*, as they are no adjectives. As our MLN was trained with object descriptions, we are likely to find other adjectives like colors in it. According to that, we determine a high similarity between our preferred word sense *yellow.s.01* and other colors and a low similarity between any other meaning and the concepts incorporated in the MLN which benefits the choice of the desired word sense.

In the context of feature comparison, the similarity plays a slightly different role. We have already committed to a certain word sense of a feature through the annotation and do not need to perform any disambiguation.

But we need a method to compare different object descriptions, preferably in allowing a certain degree of variation.

In order to do so, we developed 3 different similarity calculations, which is explained in the remainder of this subsection.

In the WORDNET lexical database, synsets are used to structure information in a taxonomy in a tree-like manner, which means, it is structured as a directed acyclic graph. Synsets are logical groupings, representing a word or collocation, containing a list of synonymous words and relational pointers to other synsets. They also allow access to information about the definition of the word as well as example sentences it may occur in, and about its POS. A part of speech may either be a noun, verb, adjective or adverb. An example of the synset for "cup" is shown in Figure 2.4

One word or collocation can be found in more than one synset and POS as it may have different meanings, while one synset may contain several different words which can be used more or less interchangeably.

To retrieve a specific meaning of a word or collocation, the respective synset can be addressed by its 3-part name of the form *word.pos.nn*, where *word* is a lemma, which is the word's morphological stem, *pos* the part of speech the word is in and *nn* the sense number, which is an id in the group of words from the respective part of speech. As an example, the word "orange" may be an adjective (orange.s.01), or it could be a noun, which can again either be the fruit (orange.n.01) or the color (orange.n.02).

WORDNET also implements different kinds of relations between synsets, represented by pointers. These relations may be either lexical (between semantically related word forms) or semantic (between word meanings). Nouns and verbs are organized in "is-a" hierarchies based on hyper- and hyponym relations with additional pointers for other relation types. It is therefore possible to retrieve all sub- or superordinate synsets (hypo-/hypernyms) of a word as well as its part-of relations to other synsets (mero-/holonyms).

Adjectives are an exception, as they are not part of the is-a hierarchy, but rather structured in clusters, each consisting of antonymous pairs of words. An antonymous pair is a pair of words with contrary meanings, such as "hot" and "cold" or "good" and "bad".

An adjective cluster contains so-called head adjectives, representing the meaning of the antonymous words. Each of those head adjectives is connected to one or more satellite adjectives, which have a similar meaning. Head and satellite adjectives can be distinguished by their POS tags, that is, a head adjective has the tag "a", a satellite adjective the POS tag "s".

The synsets of relational adjective (pertainyms) and participle adjectives again have a different structure, as they do not have antonym relations. This type of adjectives is not considered in this work. We rather focus on nouns with their respective hyper-/hyponym and part-of relations and on adjectives.

In a natural-language description, adjectives are used to specify properties of a described object. We are primarily interested in ascriptive adjectives, that is to say, adjectives ascribing a value of an attribute to a noun as introduced by Gross and Miller [8]. In the expression "The fruit is yellow", the word *yellow* would be such an adjective, as it ascribes the value *yellow* to the attribute *color* of the object *fruit*.

WORDNET provides a number of functions for determining the similarity between synsets. Most of them are path based and cannot be applied to a pair of adjectives or an adjective and a noun or verb, due to the structural characteristics of the WORDNET taxonomy described above. There are exceptions, though, which use workarounds to determine a similarity.

The problem is the semantic meaning of such a similarity calculation. It may make sense to calculate a similarity based on the position in an is-a hierarchy for objects. For example, it is intuitionally understandable that a cup is similar to a container, as they have a direct hypernym-hyponym relation, but not very similar to a fruit.

For adjectives, this relation is not entirely suitable. As an example, the adjective cluster for colors is structured based on the antonymous pair "chromatic" and "achromatic". The chromatic colors occurring in WORDNET are organized as satellite adjectives around the head adjective "chromatic.a.03", while the head adjective "achromatic.a.01" is related to various kinds of black, white and several shades of gray. This structure provides no hierarchy of the objects, but can rather be thought of as wheels, connected through an axle (the antonym relation), with the head adjectives as hubs and the satellite adjectives as spokes. An illustration of this structure can be found in Figure 2.5.

WORDNET provides a number of similarity calculations, of which one is the WUP[3] similarity. It calculates the similarity of two word senses based on their respective depths in the taxonomy and that of their Least Common Subsumer (LCS), namely, their most specific ancestor:

$$WUP(x,y) = \frac{2 \cdot depth(LCS(x,y))}{depth(x) + depth(y)} \tag{2.8}$$

The WUP similarity is therefore *path-based*, which means, that it only uses the distances between two symbols following the relational paths of the taxonomy to determine their relatedness. The semantics of this type of similarity is therefore exclusively subject to the semantic meaning of the relations represented by these paths and does not incorporate further knowledge.

In the case of the WUP similarity, the semantics is based on their relatedness based on their common ancestor. Using a path-based calculation like this would result in an identical value for each color of one wheel, that is,

$$similarity(x, yellow.s.01) = similarity(x, blue.s.01) = \dots$$

where x is another WORDNET concept (e.g. a color as well) which is connected to the head adjective of the colors through some relation. This characteristic is very counterintuitive, as one would expect to be able to differ the respective colors and for example, consider "yellow" to be more similar to "orange" than to "blue".

It shows impractical to let the similarity only depend on their taxonomic relatedness without further consideration of their semantic meaning. The relations of symbols in the taxonomy are not necessarily semantically meaningful, and therefore lead to a misinterpretation of the term "similarity". In that case, it is necessary to add further knowledge about visual perception to make an appropriate statement about their actual semantic relatedness. We introduce different measures to approximate a more human-like understanding of the similarities of objects and features in the remainder of this chapter.

Colors

Using the natural characteristics of colors suggests itself, which leads to the idea of using their respective HSV values to identify their relatedness. The representation of the HSV color model is very closely related to the human color vision and therefore suitable for adding semantic information about the similarity of the different colors.

We introduce an assignment of value triplets (*color: (Hue, Saturation, Value)*) to color symbols, which is representative for the respective color range and looks simi-

[3] Similarity introduced 1994 by Martha Palmer & Zhibiao Wu (Palmer and Wu [25]) which is used in WORDNET

lar to the representation in Table 2.2.

Color name	H	S	V
pink.s.01	335	87	87
...	...	...	...
blue.s.01	235	87	87
cyan.s.01	150	87	87
...	...	...	...
green.s.01	115	87	87
...	...	...	...
yellow.s.01	50	87	87
orange.s.01	20	87	87
...	...	...	...
red.s.01	0	87	87

Table 2.2: Feature vectors for colors: each color symbol is assigned a vector containing the respective values for the corresponding color in the HSV color model.

Note that for chromatic colors, the values mostly differ in the hue. Achromatic colors like white, black and gray on the other hand are easier to differ looking at their saturation and value. Therefore we propose using chromatic and achromatic colors separately. The similarity between one chromatic and one achromatic color is defined as a fixed value, while the similarity between two colors of the same type is calculated using their assigned values.

The Euclidean distance between each value pairs results in a lookup table containing an intuitive and human-like interpretation of their similarity, that is, yellow being more similar to green than to light-blue and even more similar to orange. The hue (H) of the HSV model ranges from 0 to 360, while the saturation (S) and value (V) lay in the range $[0, 100]$ (or $[0, 1]$). As the hue can be viewed as arranged in a circle (see Figure 2.6), we need to incorporate the fact that the two maximally dissimilar colors are not the minimum and maximum values of the range, but lay on opposite sides of the circle. In other words, the maximum difference of two hues is supposed to be 180. Otherwise, the color pink with a hue close to the maximum of the range (around 335, see Table 2.2), would be considered highly dissimilar to red (hue around 5), which does not reflect an intuitive estimation of their similarity.

The calculation is carried out as

$$sim_{color}(x, y) := \|\vec{x} - \vec{y}\| \tag{2.9}$$

where

$$\|\vec{x}-\vec{y}\| = \begin{cases} \sqrt{(x_H-y_H)^2+(x_S-y_S)^2+(x_V-y_V)^2} & \text{if } |(x_H-y_H)| <= 180, \\ \sqrt{(f(x_H)-f(y_H))^2+(x_S-y_S)^2+(x_V-y_V)^2} & \text{otherwise.} \end{cases} \quad (2.10)$$

Here, $f(x)$ is defined as

$$f(x) := (x+180) \mod 360.$$

The Euclidean distance ($\|\cdot\|$) has been slightly modified here, by introducing $f(x)$. $\vec{x}$ is the feature vector for the object *x*, which contains the HSV values in this setting.

Likewise, sizes, shapes and other properties need to be considered in some sort of hierarchy, with respect to their geometric or other discriminative characteristics.

Sizes

Sizes are a bit easier to compare as they can be organized in an ascending order. We assign a numerical value *n* in the form *size: (n)*, accordingly (see Table 2.3).

Size name	n
small.a.01	3
...	...
medium-sized.s.01	5
large.a.01	7
...	...

Table 2.3: Feature vectors for sizes: each size symbol is assigned numerical value according to its position in an ascending order from small to large.

Again, a lookup table using the Euclidean distance of the respective values is created.

$$\begin{aligned} sim_{size}(x,y) &:= \|x_n - y_n\| \\ &= \sqrt{(x_n-y_n)^2} \end{aligned} \quad (2.11)$$

Shapes

Shapes are more complicated to compare, as it is disproportionally more challenging to find an appropriate similarity measure. In this work, shapes are assigned a feature

vector containing numerical values representing their visual characteristics, such as the number of edges, angles and faces (separating 2D from 3D shapes) and a value for *subjective similarity*.

By assigning a numerical value to each shape we define an order over the different shapes which does not exist naturally. Two shapes that are considered similar therefore have a small difference between their respective *subjective similarity* value.

We define the subjective similarity in a way that it More formally, the set of features is defined as $F = \{|edges|, |angles|, |faces|, subjSim\}$, where $|x|$ denotes the number of x. An excerpt of the shape features shows a selection of the assignments of the form *shape: (|edges|, |angles|, |faces|, subjective similarity)* (see Table 2.4).

Shape name	|edges|	|angles|	|faces|	subjective similarity
...	...	...	...	...
egg-shaped.s.01	0	0	1	8
pear-shaped.s.01	0	0	1	9
...	...	...	...	...
round.a.01	0	0	1	7.5
ringlike.s.01	2	0	1	4
...	...	...	...	...
octangular.a.01	8	8	1	14
...	...	...	...	...
rectangular.s.01	4	4	1	18
...	...	...	...	...
boxlike.s.01	12	24	6	19
...	...	...	...	...
triangular.s.01	3	3	1	25

Table 2.4: Feature vectors for shapes: each shape symbol is assigned a vector containing numeric values representing geometric characteristics.

The name of a shape does not always indicate clearly, if it is two- or three-dimensional, which makes it somewhat unclear how to decide the number of edges, angles and faces. Similarly, some shapes are not well-defined at all (How many edges and angles does a star-shaped object have?) which requires some compromises in a way that the feature values are approximated to one specific instance or an (intuitively) similar shape. As an example, all shapes of the type "-angular.s.01" are considered two-dimensional and are ordered according to their respective number of angles. The features for "ringlike.s.01" are not easy to determine, as the addendum "-like" labels it as a vague description of the term "ring". Therefore its feature vector is designed to rank it somewhere near roundish shapes without sharp edges and angles.

The result (calculated again using the Euclidean distance) is a lookup table that

represents some sort of ranking, in which rectangular is considered more similar to pentagonal than to hexagonal or septagonal and crescent to be more similar to round than to cuneate or conic. This configuration has shown useful for this purpose, but can be modified taking more characteristics into account.

$$\begin{aligned} sim_{shape}(x, y) &:= \|\vec{x} - \vec{y}\| \\ &= \sqrt{\sum_{f \in F} (x_f - y_f)^2} \end{aligned} \tag{2.12}$$

Another observation is the maximal dissimilarity of the instances of these property types among each other. Colors are maximally dissimilar to everything, that is *not* a color, just as shapes are maximally dissimilar to everything, that is *not* a shape as they intuitively have nothing in common.

An illustration of how individual branches of the WORDNET taxonomy are used for a separate similarity calculation is shown in Figure 2.7.

Hypernyms

Following the descriptions above, the similarity calculation makes use of predefined similarities for shapes, colors and sizes. But what about the rest?

The WORDNET taxonomy is based on hyper-/hyponym relations between the synsets, which we can take advantage of. We make the assumption, that if one synset occurs in the hypernym path of another synset, they may be closely related, as they have something in common.

Also, the smaller the distance between the two concepts, the higher is the expected similarity. Following the observation, that the specificity of the concepts increases, the further down they are located in the sub-tree, we further assume that two concepts that are very far from the root, and therefore highly specific, are more closely related than two concepts with the same distance closer to the root.

In Figure 2.8 we can see, that two concepts on the same sub branch close to the root do not necessarily need to be very similar. As an extreme example, the concept *entity.n.01* (which is the root of the WORDNET taxonomy tree) describes the class of *everything*, while its direct ancestor *physical_entity.n.01* comprises only "touchable" things. The classes are therefore in a rather superficial hyper-/hyponym relationship.

Further away from the root, with increasing specificity, the concepts become more similar to each other. The concepts *cup.n.01* and *coffee_cup.n.01* are almost identical, and only differ in their respective specified purpose. Their long list of common ancestors indicates a close relationship between them.

The calculation of the hypernym relationship of two concepts therefore relates their distance to each other to their maximum depth to the root:

$$sim_{hyp}(x, y) := 1 - \frac{|depth(x) - depth(y)|}{max(depth(x), depth(y))} \tag{2.13}$$

In this calculation, *depth(x)* denotes the depth of the synset *x* in the taxonomy (i.e. its distance to the root), $|\cdot|$ the absolute value. The function *max(*x_0*,*x_1*,...,*x_n*)* returns the maximum value of the arguments x_0 to x_n.

WUP

If the synsets we want to compare are not hyper- and hyponym, respectively, we use another way to calculate the similarity. As mentioned previously, adjectives are not structured in an is-a hierarchy, which is why most similarity functions provided by WORDNET can not handle them at all, or not sufficiently.

There is, however, a possibility to avoid using the adjective itself for the similarity calculation. Each synset represents a sense of a word form (lemma), which holds an attribute called "derivationally related forms", representing semantically related terms from different syntactic categories (e.g. nouns). The derivationally related form of the adjective "yellow.s.01" is "yellow.n.01", which is the noun of the color yellow, as we can see from its description "yellow color or pigment; ...". We can therefore consider it a suitable substitute for the adjective. As adjectives are structured in separate clusters, their lemmas' derivationally related forms are the link to the actual WORDNET taxonomy, as you can see from the visual representation in Figure 2.9.

A modified version of the WUP similarity introduced earlier (Equation 2.8) is used for the nouns related to those adjectives.

The similarity of an adjective to another word would be calculated using its semantically related term, instead of the adjective itself. As a result, there is a difference in the semantics of the two similarities, because the two terms to be compared are treated as if they were identical, which they are not. To avoid this counterintuitive treatment, a factor (*pFactor*) to "punish" the transformation to another term is introduced:

$$WUP_{mod}(x, y) = \frac{2 \cdot depth(LCS(x, y))}{depth(x) + depth(y) + pFactor} \tag{2.14}$$

This value is increased by a fixed value for each synset that is compared and has to be "transformed" to another synset.

Another observation of the taxonomies' characteristic comes into play now: entities that are located near the root are usually not very similar to each other, as they represent highly general superordinate classes of the elements deeper in the taxonomy

tree. Even if two elements near the root have the same path distance to each other as two elements deeper in the taxonomy, their relatedness is most likely much less significant.

To mimic this effect, the punishment factor mentioned above is not only increased for the adjective transformation, but also when the two synsets to be compared are located in different taxonomy branches.

Figure 2.10 shows an example of what the depth of the split of a branch reveals about the similarity of two concepts in the created sub-branches. As mentioned before, the specificity of the concepts increases, the further down in the taxonomy they are located. A direct consequence is, that a branch that was split closely to the root results in two separate branches of which the concepts become more and more specific, independent on the respective other. If, on the other hand, a branch was split very low in the taxonomy tree, the concepts of the two sub-branches have a long list of common ancestors and therefore more in common than the ones whose branches were separated close to the root.

Analogously to the example in Figure 2.8, the two concepts *coffee_cup.n.01* and *paper_cup.n.01* are considered very similar to each other, while their superordinate concept *cup.n.01* is rather dissimilar to the concept *cargo_container.n.01*, although the distances through their respective ancestors are the same as for *coffee_cup.n.01* and *paper_cup.n.01*.

In this calculation, it is taken into account, *where* the split of the branches took place, that is, the punishment factor in Equation 2.15) is increased inversely proportional to the depth of the split. The new punishment factor *pFactor'* is determined by adding the distance of the two concepts under consideration of the depth of their common ancestor to the old punishment factor *pFactor*.

$$pFactor' := pFactor + \left(1 - \frac{depth(LCS(x,y))}{max(depth(x), depth(y))}\right) \tag{2.15}$$

The modification allows to adapt to several characteristics of the inference process and adds more semantic knowledge to the similarity calculation.

Overall Similarity

All of the similarity calculations introduced above are heuristics to avoid the purely taxonomic relatedness of the exclusively path-based similarities. Summing them up, we distinguish 3 different similarity calculations:

- The Euclidean distance of predefined values, used for colors, sizes and shapes
- The special case for hypernym relations
- A modified version of the WUP similarity

On that account, the overall similarity calculation of the properties is calculated as follows:

$$sim(x,y) := \begin{cases} 0 & \text{if } is\_a(x,type) \oplus is\_a(y,type), \\ sim_{type}(x,y) & \text{if } is\_a(x,type) \wedge is\_a(y,type), \\ sim_{hyp}(x,y) & \text{if } x \in HPath_y \vee y \in HPath_x, \\ WUP_{mod}(x,y) & \text{otherwise} \end{cases} \tag{2.16}$$

The function *is_a(x,y)* returns true, iff *x* is of type *y* with *type* $\in$ {color, size, shape}. The operator $\oplus$ represents the logical operator for exclusive disjunction (XOR). $HPath_x$ is the transitive closure of the hypernyms of *x*.

According to the methods described above, further (predefined) similarities can be specified, based on the respective characteristics of the feature types. It is an easy and intuitive way to define similarities on those features, that can be arranged in some sort of order or taxonomy, but other definitions may work as well.

2.5 MODELING MLNS - AN EXAMPLE

In Section 2.2 we mentioned that in order to reduce complexity, we convert the results of the transformation process (Part I) to a different representation In particular, we use a separate predicate for each property type instead of only one.

In the following, we will explain this decision by means of an example.

There are different possibilities how MLNs can be modeled of which some may look equivalent at first sight. As a reminder, an MLN is represented by first-order logic formulas and their respective weights. Each formula consists of atoms (predicate symbols with their arguments) connected by logical operators like $\wedge$ (AND), $\vee$, (OR), or $\neg$ (NOT). The arguments of an atom may be typed, depending on the domains defined for the predicate.

The variables *a* and *b* of the atom *example(?a,?b)* both need to be of type *word* (i.e. both are in the domain *word*) if the predicate is defined as *example* $\subset$ *word* $\times$ *word* (which looks like *example(word, word)*). If it is defined as *example* $\subset$ *word* $\times$ *sense*, only *a* can be of type *word*, as *b* needs to be of type *sense*. The following example describes the impact of different typing strategies.

Below, two different modelings of the MLN used for the object inference are described and compared through an example. The first one (denoted as modeling I in the following) uses the predicates

$$size \subset cluster \times size,$$

$$color \subset cluster \times color,$$

$$shape \subset cluster \times shape,$$

$$hypernym \subset cluster \times hypernym)$$

and

$$hasa \subset cluster \times hasa$$

to represent the attributes of an object while the second one (modeling II) uses only one predicate

$$property \subset cluster \times word \times \{COLOR, SIZE, SHAPE, HYPERNYM, HASA\}$$

additionally to the predicate

$$object \subset cluster \times objID$$

respectively. The result is a separate domain for each property variable (*size, color, shape, hypernym, hasa*, respectively) instead of one large shared domain (*word*) which combines all property values regardless of their respective type.

The advantage lays in the number of generated ground formulas from the formula template and accompanying decreased computation time. As an example, assume the formula

```
object(?c, +?objID) ^ color(?c, +?w)
            ^ size(?c, +?w)
            ^ shape(?c, +?w)
```

Listing 2.12: Formula Template for modeling I

for modeling I and equivalently,

```
object(?c, +?objID) ^ property(?c, +?w, COLOR)
            ^ property(?c, +?w, SIZE)
            ^ property(?c, +?w, SHAPE)
```

Listing 2.13: Formula Template for modeling II

for modeling II.

These formulas are again constructed using *PRACGrammar*, introduced in Section 2.3 (Constants are written in capitals).

The number of generated formulas from the template of modeling I, with each property being represented through an own predicate, amounts to

$$o \cdot \prod_{pa \in P_f} d_{pa} \tag{2.17}$$

where *o* is the number of elements in the *objID* domain (i.e. $o = |objID|$), P_f is the set of all occurring property atoms in the formula template (which is $P_f = \{color, size, shape\}$ in this example) and d_{pa} denotes the number of elements in the domain for the given property variable in the atom (i.e. $|size|$, $|shape|$, ... respectively).

In modeling II the number of generated formulas is much higher through the shared *word* domain:

$$o \cdot w^{n_f \cdot p} \tag{2.18}$$

where *w* is the number of elements in the *word* domain ($w = |word|$), n_f is the number of the property-Atoms occurring in the formula template and *p* is the number of elements in the *prop* domain ($p = |prop|$).

Note that

$$\bigcup_{p \in P} p = word \tag{2.19}$$

where *P* denotes the set of all possible property domains, i.e. $P = \{color, size, shape, hypernym, hasa\}$.

Some elements from the domain *word* from modeling II may be in one or more domains from modeling I (a described object can either *be* a container (i.e. $container \in hypernym$) or *have* one (i.e. $container \in hasa$)), which means that they are not disjoint sets of words.

To see the difference in the two formula modelings, consider the following example (see Table 2.5): If there is a dataset containing 6 different objects ($|objID| = 6$) and 9 different property values (i.e. $|word| = 9$) the formula template in Listing 2.13 would be grounded to

$$|objID| \cdot (|word|)^{\#atoms \cdot p} = 6 \cdot 9^{3 \cdot 1} = 4374$$

different formulas, according to Equation 2.18. Note that *p* equals *1* here, as the formula template already contains constants from the *prop* domain instead of a variable with a prefixed "+" to expand every single value of it.

Modeling	**|objectIDs|**	**|property Values|**	**property domain type**	**|formulas generated|**
I	6	9	separate	144
II	6	9 (distributed on separate domains)	shared	4374

Table 2.5: Comparing modelings: This table shows a significant difference in the number of generated formulas depending on the use of one shared or multiple separate property domains

We assume the same number of objects for the multiple predicate modeling I. The 9

elements from the *words* domain are split up to the three separate domains (color: 4 elements, size: 3 elements and shape: 2 elements).

The equivalent formula template Listing 2.12 would be grounded to only

$$|objID| \cdot |color| \cdot |size| \cdot |shape| = 6 \cdot 4 \cdot 3 \cdot 2 = 144$$

different formulas according to Equation 2.17.

The number of formulas is further reduced drastically here, if one of the property domains is empty, which makes sense, as no statement can be made about the correlation of the three property attributes and the respective object. In the example described here, no valid formula would be grounded from the formula template if, say, the *shape* domain was empty, as $|objID| \cdot |color| \cdot |size| \cdot |shape| = 6 \cdot 4 \cdot 3 \cdot 0 = 0$. For comparison: in modeling II, still $|objID| \cdot (|word|\text{-}\#shapes)^{\#atoms \cdot p} = 6 \cdot (9\text{-}2)^{3 \cdot 1} = 2058$ formulas are generated, as the *word* domain is simply reduced by the number of elements previously in the *shape* domain.

Not only the numbers of the generated formulas differ in the two models, but also their semantic meaning may vary. Most of the *4374* generated formulas from Listing 2.13 do not make much sense, as it is not possible to differ between the types of properties.

```
0.000000    object(?c, spoon.n.01) ^ prop(?c, round.a.01,
   SHAPE) ^ prop(?c, concave.a.01, COLOR)
0.000000    object(?c, spoon.n.01) ^ prop(?c, small.a.01,
   SHAPE) ^ prop(?c, curved.a.01, COLOR)
```

Listing 2.14: Example of generated formulas

Formulas like the ones in Listing 2.14 are created, which end up with a formula weight of *0* in the learning process (meaning no statement can be made about the truth of this formula), because the attributes *concave* and *curved* are obviously no colors but shapes, while *small* is not a shape but a size attribute. Nevertheless, the formulas have to be created as all these attributes belong to the *word* domain and are expanded according to the formula template in the grounding process.

The calculation above only exemplifies the generation of ground formulas from one formula template. In general, there is more than one formula template required to create a representative model. This example shows, how allegedly similar modelings of an MLN can cause different results which may vary to a great extent in the number of generated formulas and subsequently, computation time.

Chapter 4 deals with the qualitative analysis of both approaches and compare their results when applied to actual test data.

2.6 SUMMARY

In the previous sections we introduced how natural-language descriptions can be used in object recognition tasks in a way that they replace actual instances of training examples. The question we want to answer is, if it is possible to recognize objects only from their respective NL descriptions without using any training examples at all.

We avail ourselves of additional knowledge about the structure of natural language and taxonomic relation provided by the Stanford parser and WORDNET to render a transformation from NL descriptions to a formal representation possible. The NL descriptions can be retrieved from the internet and processed by means of word disambiguation and similarities between concepts. Using information about the structure of natural language and taxonomy thereby adds common knowledge to a certain extent to the object recognition task. The result is a formalized object description with the property attributes extracted from the NL phrase.

The requirement of numerous instances of the objects to train a knowledge base is therefore not present anymore. Another advantage of this approach lies in the generality of the model. NL object descriptions from online dictionaries usually describe the general concept of an object, which leads to a more general model than the one generated using a conventional approach. This is due to the fact that training a model with example instances results in a model representing the attribute-object relations from the training data, rather than the general concept of the object classes.

To be able to represent a joint probability distribution over the observations, an MLN is used to realize the model. Based on our findings in the comparison of different modelings we conclude that allegedly similar formula templates may result in different MLNs and therefore produce different inference performances. A number of modelings and configurations are designed and evaluated to find the most suitable solution.

We explained that using only the taxonomic relations in WORDNET is not adequate to define a similarity between concepts, as a taxonomy-based similarity does not necessarily reflect a natural understanding of object relations. As a consequence, we developed individual similarity calculations for different object types, and methodologies how to interpret the semantic relations in the taxonomy in a more intuitive way.

In the next chapter, we continue with our implementation of the inference pipeline.

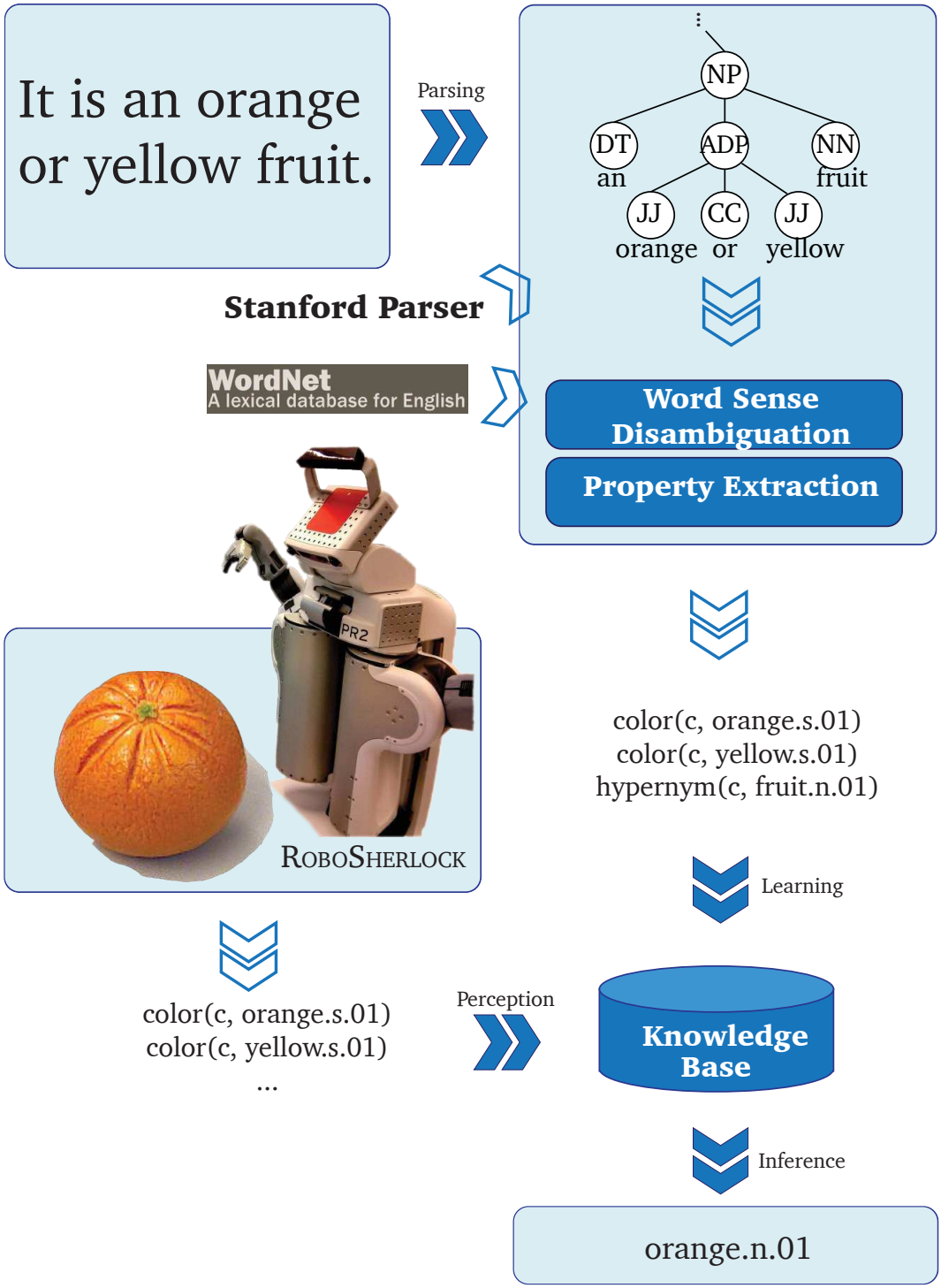

Figure 2.3: Pipeline revisited: The word sense disambiguation is executed on parsing result using information from WORDNET. The knowledge about the structure of natural language is used to extract the property attributes from the description. The resulting formalized object description is used to train a model which can be used as a knowledge base for inferring an object's identity from the perception.

Synset('cup.n.01'):

- **pos:**
  - 'n'
- **name:**
  - 'cup'
- **definition:**
  - 'a small open container usually used for drinking; usually has a handle'
- **hypernyms:**
  - [Synset('container.n.01'), Synset('crockery.n.01')]
- **hyponyms:**
  - [Synset('teacup.n.02'), Synset('grace_cup.n.01'), Synset('coffee_cup.n.01'), Synset('dixie_cup.n.01'), Synset('chalice.n.01'), Synset('mustache_cup.n.01'), Synset('beaker.n.02'), Synset('kylix.n.01'), Synset('scyphus.n.01')]
- **...**
  - ...

Figure 2.4: Synset Info: Example of the information, a synset contains about the represented object.

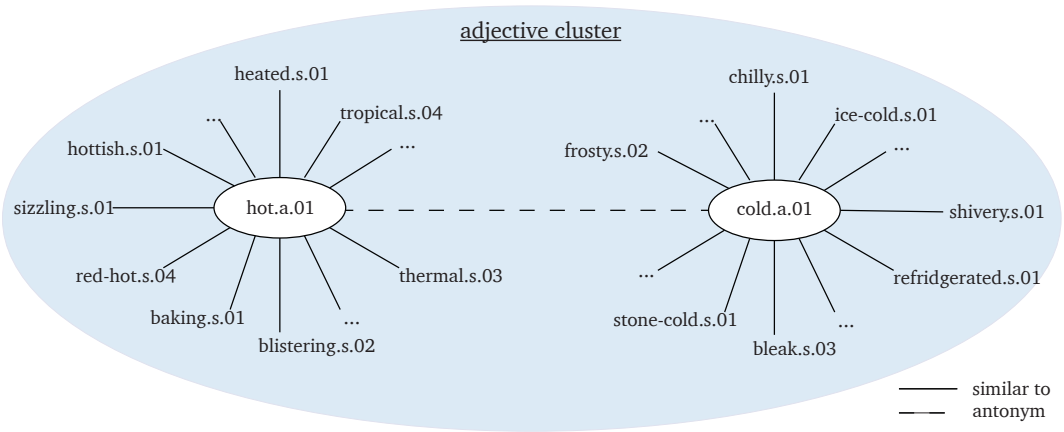

Figure 2.5: Adjective cluster: the two head adjectives (hot.a.01 and cold.a.01) are connected to each other through an antonymy relation, and to their respective satellite adjectives through similar_to relations.

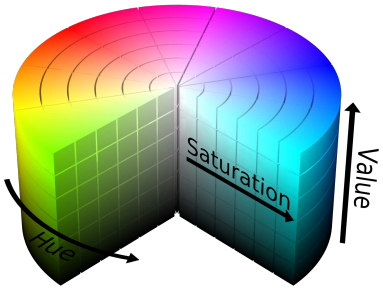

Figure 2.6: HSV Cylinder: The HSV color model can be represented as a cylinder, the color values arranged along the axis, the saturation along the radius and the hue along the angles of the base.

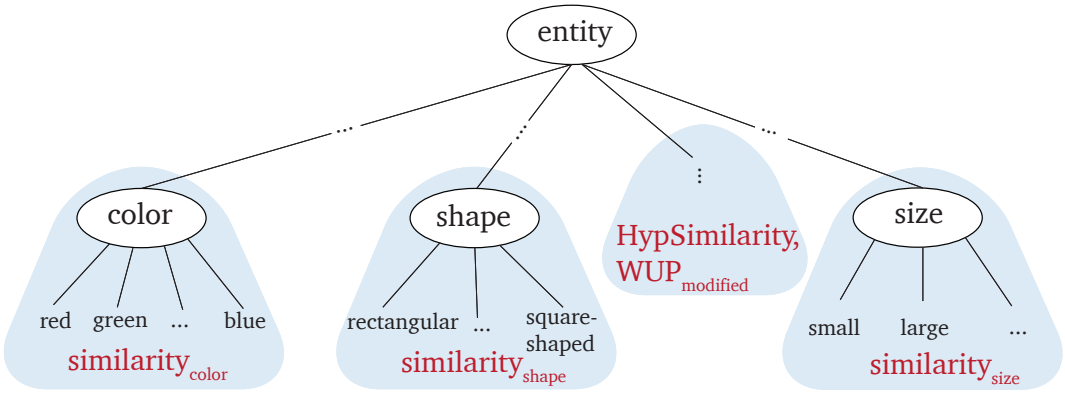

Figure 2.7: For the categories shape, size and color there exist individual similarity calculation, for the rest of the taxonomy we use the similarity calculation based on their taxonomy relations.

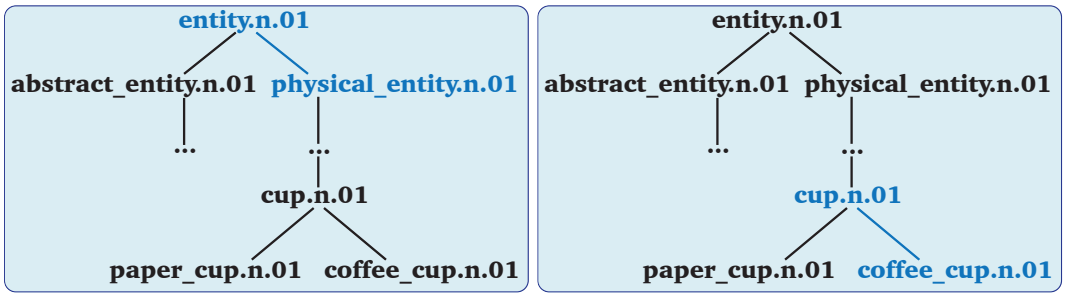

Figure 2.8: Hypernym relation: If one synset is found in another's hypernym paths, they are assumed closely related. The degree of relatedness is dependent on their distance to a common ancestor and its distance to the root.

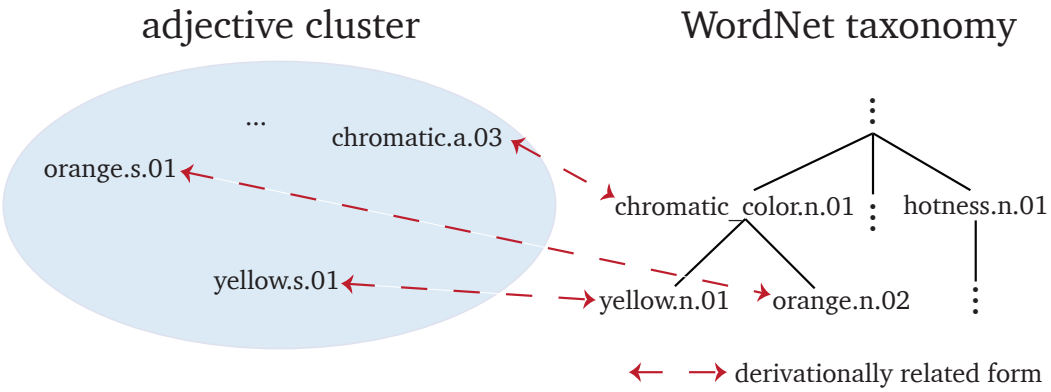

Figure 2.9: The adjectives in the cluster (blue) on the left are connected to other objects in the taxonomy (right) only through their derivationally related forms.

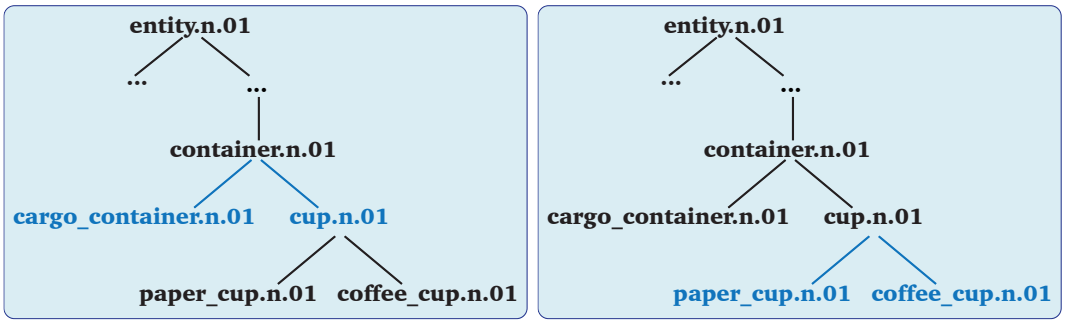

Figure 2.10: Taxonomy branch relation: In a taxonomy with semantic relations between the nodes, concepts are more similar to each other, the further their common ancestor is away from the root node, i.e. when the split of the two taxonomy branches of the concepts is very low in the taxonomy tree. Concepts close to the root are rather general classes and do not necessarily be closely related to concepts on the same level.

CHAPTER three

IMPLEMENTATION

In this chapter, the implementation of the developed system is outlined. An introduction on how to execute the inference pipeline is given in Section 3.3.

The implementation employs the Probabilistic Robot Action Cores (PRAC) system, introduced by Nyga and Beetz [22], which is used to model abstract event types learned from natural-language instructions. This is to equip autonomous robots with action-specific knowledge to be able to perform complex everyday activities. The system provides tools for statistical relational learning and reasoning, as well as evaluation and incorporates MLNs and probabilistic first-order knowledge bases. By using these knowledge bases it is possible to infer missing information from the original natural-language instructions by using generic event patterns.

The PRAC system is a framework for iterative reasoning. It is structured in a way that is uses *modules* for learning and inference tasks. Each module performs one or more database transformations. Multiple modules can be executed consecutively to create a database transformation pipeline in which each module takes the output database of the previous one as its input.

We realize the first two core parts (Subsection 2.4.1 and Subsection 2.4.2) by implementing such *modules*. In doing so, we apply PRAC on the extraction of object characteristics instead of event types.

Each of the modules serves different purposes, respectively; one for the transformation of natural-language descriptions into a formal, logic-based representation and one for inferring the most probable object identity from the object attributes.

3.1 FEATURE EXTRACTION - LEARNING AND INFERENCE

To train the MLN, a dataset consisting of separate databases representing object descriptions is used. They are generated from NL object descriptions, but render no connection between the description and the object itself. They rather represent the relation between the grammatical structure of natural language and the information contained in it. To obtain a vast range of possible properties, descriptions of various objects, which do not necessarily match the ones that should be recognized later, have been used.

Each database has been annotated with syntactic evidence of the NL description from the Stanford Parser and WORDNET annotations to match the correct senses to the words in the description. The object properties contained in the natural-language description as well as missing word sense information are annotated manually.

The databases are used for a supervised training of a Markov Logic Network, which is used to model the transformation of a natural-language description to a formal first-order logic representation. The MLN is learned using pseudo-log-likelihood optimized using direct descent with a learning rate of 0.9.

For the inference step, some preprocessing of the evidence database is made to add similarity information about the words used in the input description and the ones already incorporated in the MLN.

This is to ensure that, even if a word from the description is not known from the training data, it can still be identified as a potential property through its semantic similarity to other concepts.

To infer the most probable word senses as well as the properties, the MLN is transformed into a WCSP, to estimate an efficient MPE.

3.2 OBJECT RECOGNITION - LEARNING AND INFERENCE

The learning step for the object inference uses the semantic attributes size, shape, color, hasa and hypernym to train the model.

The training data consists of formal object descriptions generated from natural-language descriptions obtained from WORDNET[1], WIKIPEDIA, FREEBASE[2], WIKTIONARY[3] and the ENCYCLOPÆDIA BRITANNICA[4].

[1] http://wordnetweb.princeton.edu/perl/webwn
[2] https://www.freebase.com/
[3] http://en.wiktionary.org/
[4] http://www.britannica.com/

The training data is created from the content of the NL descriptions found on these websites, therefore they only contain semantic attributes and no low-level features.

Just as in Section 3.1, pseudo-log-likelihood direct descent optimization with a learning rate of 0.9 is used.

For the inference step, again, some preprocessing is required. Inference is not only based on these strict definitions, but allows a certain degree of interpretation. As humans tend to use different variations of descriptive terms, which are not identical but rather similar, this information is added in the inference process to allow an object to be recognized even if it does not fit its description in the knowledge base completely. In terms of the implementation of this processing step, the semantic similarities of words are used to identify the degree of difference between two object descriptions.

Not only word similarities are added as described before, but also a database transformation is executed. As the result from the inference step described in Section 3.1 is the evidence for the inference step described here, and the first MLN is modeled using only one property predicate, the evidence needs to be converted to fit the model used for the object inference.

To serve the purpose of reducing complexity (see Section 2.5), we require the evidence to be in the form described in Listing 2.12. The inference itself is again executed by transforming the problem into a WCSP and estimating its MPE.

The implementation of the modeling described in Listing 2.13 adds a closed world assumption implicitly by having to append all possible word-property combinations when adding the word similarities, due to the disadvantageous usage of only one shared *word* domain. This circumstance is explained in more detail in the evaluation.

3.3 USAGE

Within the scope of this work, two main modules have been implemented to infer an object from a given natural-language description. The module *prop_extraction* is used to extract information from a given expression, that is, it infers properties such as size, shape, color and superordinate terms (hypernyms) of the described object. A trained MLN representing knowledge about the structure of natural language is used to analyze the NL expression and retrieve the desired object properties from it.

The second module, *obj_recognition*, uses this information to infer an object that matches the specification. To do so, it uses a knowledge base, which is basically an MLN defining objects in terms of rules, that conjunct the properties of the objects, respectively.

It can be constructed from a database by calling

```
praclearn --dbs /path/to/trainingdata.db --mln /path/to/
   formulatemplates/filename.mln FirstOrderLogic --module
   modulename
```

Several db files may be listed, separated by comma, to train from multiple databases. The *modulename* may be either *prop_extraction*, *obj_recognition* or *-_old*. An MLN is generated containing all groundings for each formula for each object in the database in */path/to/formulatemplates/*, named *filename_trained.mln* (according to the filename of the given formula templates).

To infer an object's identity from a given natural-language description, run

```
pracobjrec -i 'object description'
```

A GUI tool will be started in which the modules can be loaded.

To run the property extraction on the object description select the module *prop_extraction* and the Knowledge Base *default* (see Figure 3.1). The fields for the options *Logic*, *MLN*, *Method*, *Queries* and *Parameters* are filled automatically with the default settings, the field *Evidence* contains syntactic information of the sentence (parameter *object description*).

The button *Start Inference* triggers the execution of the selected module. With the button *Continue >* the results of the module's database transformation can be stored in the evidence field. This can be used to run several modules in a pipeline as another module can now be executed using the updated evidence.

The evidence field can also be edited manually or filled with the content of previously stored evidence files.

To infer the object identity from a given object description select te module *obj_recognition* (or *obj_recognition_old*, respectively). A previously trained MLN can now be loaded in the dropdown box *MLN* and executed as described above (see Figure 3.2).

When closing the program a formatted result is printed in the standard output.

To run the object inference module, an example query similar to the following may be inserted into the *Evidence* field:

```
hypernym(c,cutlery.n.02)
size(c,shallow.a.01)
shape(c,bowl-shaped.s.01)
hasa(c,handle.n.01)
```

Listing 3.1: Description for "spoon.n.01"

It is also possible to run the modules *prop_extraction* and *obj_recognition* (or *-_old*) in a pipeline:

```
pracobjrec -r -m /path/to/filename_trained.mln FuzzyLogic
   'object description.'
```

The *filename_trained.mln* is the MLN file learned using *praclearn* (see above). Adding the optional flag *-o* calls the module using the modeling described in Listing 2.13.

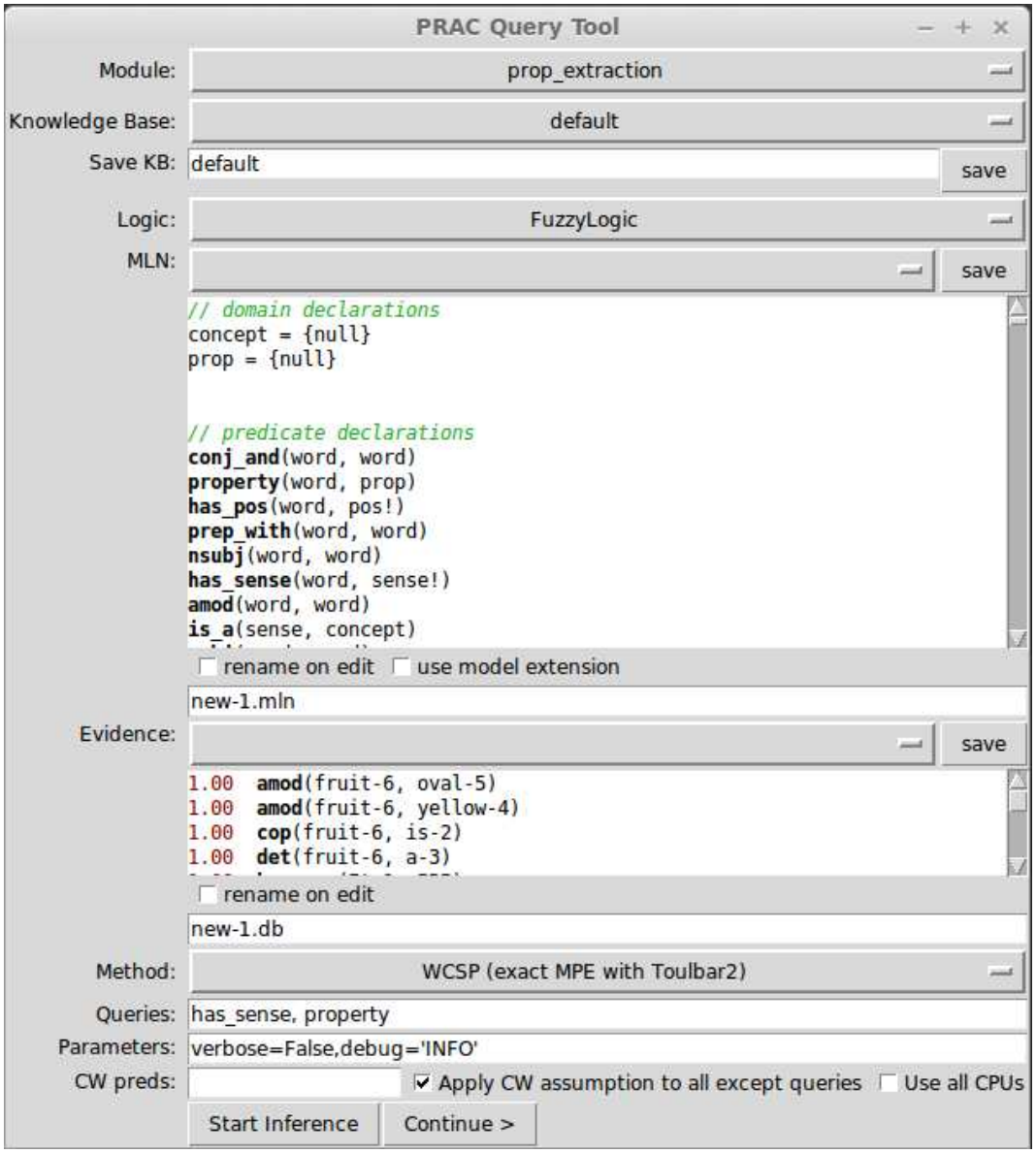

Figure 3.1: PRAC Query Tool for the inference pipeline. The field for the evidence is automatically filled with the syntactic evidence from the parsed object description in the parameter.

The sentence *object description* is parsed and transformed into a formal description, which is automatically used as evidence for the second module.

The result is the inferred object identity.

To execute a k-fold cross-validation on the *obj_recognition* module, run:

```
pracxfold --dbs /path/to/trainingdata.db --mln /path/to/
   formulatemplates/filename.mln --predicate object --
   domain cluster --folder evalResult --module
```

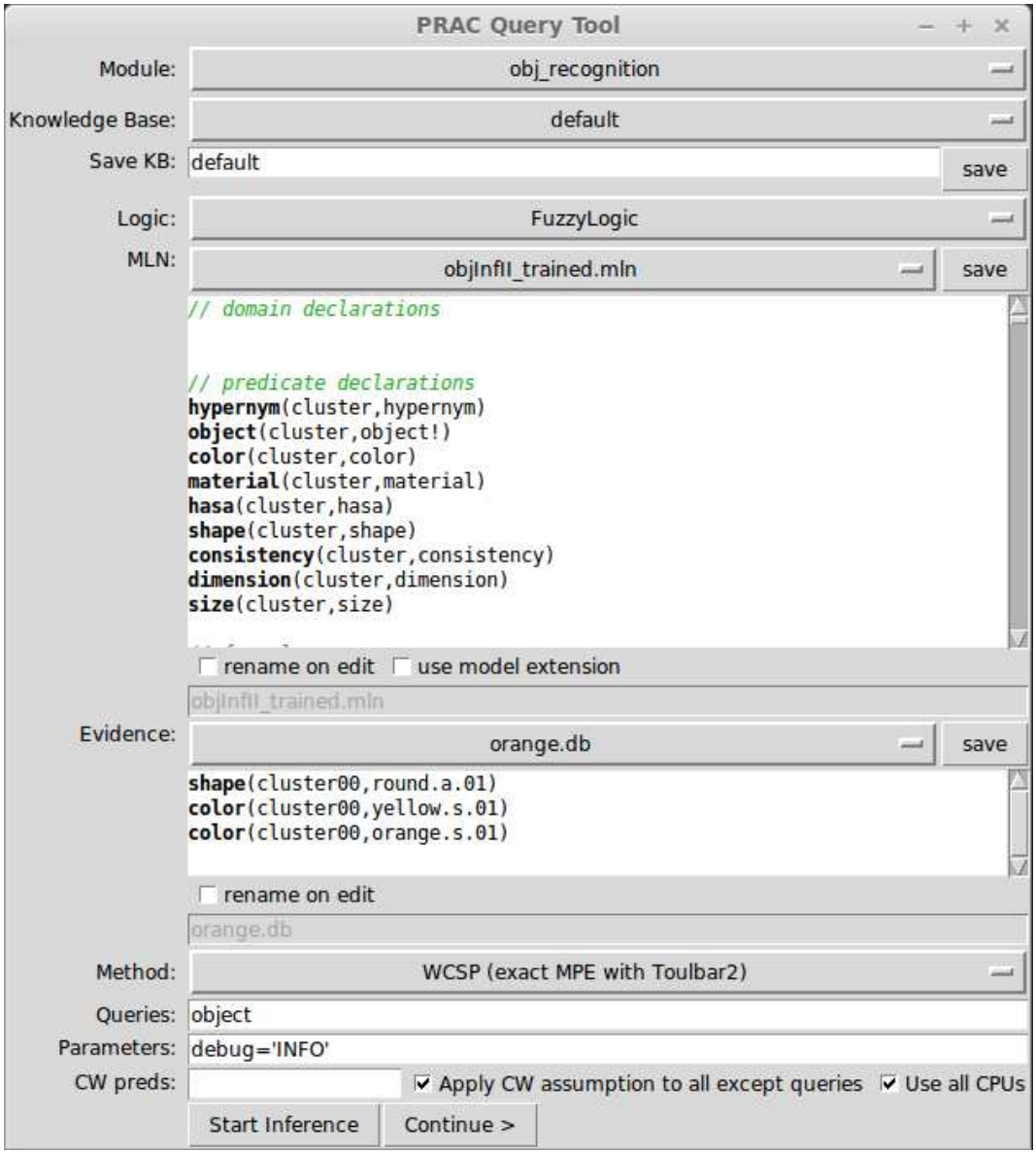

Figure 3.2: PRAC Query Tool for the inference pipeline. The evidence field contains the formal representation of the object description and is used in the *obj_recog* module to infer the object identity.

```
obj_recognition --multicore --folds 10
```

Listing 3.2: test

The module to be tested can be specified with the parameter *–module*, the number of folds for the cross-validation is declared with *–folds*.

The flag *–object* denotes the predicate that is queried, *–domain* is the domain from this predicate that is ignored. The predicate *object* for example is defined as $object \subset cluster \times objID$ and we are primarily interested in the value of *objID*, as *cluster* is only

an identifier for the cluster, not the object itself. The results of the cross-validation are stored in the folder specified by the parameter *–folder*. To run the validation using multiple cores, use the flag *–multicore*.

There is another special flag called *–altMLN*, which can be used to specify an alternative MLN to read in the training databases, if they are in a different format than expected by the module. This flag is used to be able to evaluate the module *obj_recognition_old*, as the training databases first have to be transformed into the other modeling mentioned before.

In the next chapter, we evaluate the concept using a k-fold cross-validation. The results are investigated using different performance measures. A visual representation of the results are provided in terms of confusion matrices contrasting the output class of the inference steps and their respective actual target classes.

CHAPTER four

EXPERIMENTS AND RESULTS

This chapter covers the tests that have been run on different modelings and configurations of the MLN used to infer object identities from their attributes.

4.1 SETUP

For the evaluation, we use a k-fold crossvalidation, which is a method to evaluate models especially for classification tasks. In a k-fold cross-validation, the training data is randomly partitioned into k sets of equal size. The evaluation runs k times and in each step, *k-1* of the datasets are used to train the model that is to be evaluated and the remaining one is used for testing. This happens in a way that each of the k subsets is omitted in the learning step and used for testing once. k is not fixed and can be anything between *2* and n, where n is the number of available training examples. If $k = n$, the k-fold crossvalidation is called leave-one-out crossvalidation. A commonly used value for k is *10*, which is used in our evaluation as well.

To evaluate the semantic perception model generated from natural-language descriptions, a 10-fold cross-validation on a dataset containing 56 databases representing 14 different objects is executed. For each object, there exist four to five training examples, depending on the quality of the descriptons from the respective source.

Sometimes, an object description does not contain any perceivable information, but rather describes an object with its application purpose or relation to other objects. In this work we focus on perceivable features which is why we concentrate on training data containing visual attributes which can be represented by our model.

The evaluation of the results is based on the quality measures *accuracy*, *precision*, *recall* and *F1-score*, which is explained below.

In the following, we use abbreviations to refer to certain subsets of the classified examples:

- **True Positives (TP)** are examples that have been correctly classified as positive
- Analog, **True Negatives (TN)** are examples that have been correctly classified as negative.
- **False Positives (FP)** are examples which have falsely been classified as positive (also referred to as *false alarm*).
- **False Negatives (FN)** should have been classified as positive, but were not (called *miss*)

Note, that we assume a two-class classification problem here. A multi-class classification problem can be interpreted as a two-class classification problem in a way that the classes are separated into two groups; one for the class we are interested in representing the *Positives* and one for any other classes representing the *Negatives*.

The terminology is also visualized in Table 4.1.

pred\class	**Positive**	**Negative**
Positive	TP	FP
Negative	FN	TN

Table 4.1: Terminology of the confusion matrix: Correctly classified examples are referred to as True Positives (TP) and True Negatives (TN), respectively. False Positives (FP) have erroneously been classified as positive, while False Negatives (FN) should have been classified as positive, but were not.

The *accuracy* is the fraction of correctly classified examples that is

$$\frac{TP + TN}{TP + FP + FN + TN} \tag{4.1}$$

The *precision* or *positive predictive value* is a measure denoting the fraction of the examples classified as class *x*, which actually are class *x*:

$$\frac{TP}{TP + FP} \tag{4.2}$$

The *recall* or *hit rate* is the rate of true positives, meaning how many examples from actual class *x* have been classified as *x*.

$$\frac{TP}{TP + FN} \tag{4.3}$$

The *F1-score* is the *harmonic mean* of *precision* and *recall*:

$$\frac{2 \cdot TP}{2 \cdot TP + FP + FN} = 2 \cdot \frac{precision * recall}{precision + recall} \tag{4.4}$$

The results are visualized by a representative confusion matrix of the respective crossvalidation. The rows denote the predictions of the model, while the columns represent the ground truth that is, the target class. Correct classification results (prediction = ground truth) are in boldface. The intensity of the cell color increases with its respective value.

4.2 EVALUATION

In the following sections 4.2.1 to 4.2.4, not only the two different modelings but also the results from MLNs generated from different formula templates are compared. As a quick reminder, modeling I refers to the predicate modeling using different predicates (*shape, size, color*) for different predicate types, while modeling II refers to the one using only one (*property*) predicate, as mentioned in Section 2.5.

The different formula templates are referred to as *configurations* (CONF I, II, III) and can be described shortly as follows:

- Configuration I consists of five formulas representing object-feature relations. Three of them contain two features (i.e. represent all possible combinations of *size, shape* and *color*), the remaining two contain only one (*hypernym* or *hasa*, respectively.).

- Configuration II results from observations on how objects are usually described in natural language. This is in particular the tendency to use one visual attribute in combination with a superordinate concept to describe an object. It contains the formula templates from configuration I and expands them by adding the combinations of the *hypernym* feature with all other attributes.

- Configuration III consists of only three formulas, of which one conjoins the three attributes *color, size* and *shape* and therefore assumes that all three features are used to describe an object. The remaining two formulas relate the object to its *hypernym* and *hasa* attributes.

The configurations can be looked up in the appendix (see MLN Templates - Modeling I and MLN Templates - Modeling II).

4.2.1 COMPARING MODELINGS

As already explained by means of an example in Section 2.5, different modelings of the predicates (and therefore formula templates) will result in different ground MLNs. The difference is not only evident when comparing computation time for generating the MLN, but also influences the results of future inference steps.

In the following, the two different modelings are compared using equivalent configurations. In other words, the two configurations of the formula templates look like they are expressing the same, but we will see that there is a difference in the informative values of the generated models.

The MLN generated from modeling II contains one formula for each grounding of the respective formula template. Each variable can take each value from the *word* domain introduced before, which means that we have to add all word-property combinations with the respective similarity to the evidence. As an example, if we had perceived the color *blue* for the unknown object our evidence database would contain *property(cluster, blue.s.01, COLOR)*. To add our similarity information we would have to add *property(cluster, w_i, COLOR)*, for each w_i in the same domain as *blue.s.01*, i.e. for each $w_i \in$ *word*. The truth value of this added atom would be the respective similarity of w_i and *blue.s.01*.

We therefore execute the following steps:

For each evidence value w_i of type $pType_k$:

- We assert false properties, which means that we add $\neg$*property(cluster, w_i, $pType_l$)* for each $pType_l \in prop \setminus \{pType_k\}$. Intuitively this can be interpreted in a way that if an attribute is of type $pType_k$, it can not be of type $pType_l$. In the example above we would assume that if *blue.s.01* is of type *COLOR*, it can not be a *SIZE* or *SHAPE*. This assumption is not entirely true for all cases, but it approximates the reality sufficiently.
- For each $w_j \in word$, $w_j \neq w_i$ we then add *property(cluster, w_j, $pType_k$)*.
- Analogous to the assumption before, we add $\neg$*property(cluster, w_j, $pType_l$)* for each $w_j \in word$, $pType_l \in prop \setminus \{pType_k\}$.

This happens for each evidence property, which means that in general we approximate a closed world assumption implicitly.

We therefore need to make a closed world assumption for modeling I as well to establish the same general conditions as for modeling I. This is necessary to ensure that we really compare the two different modelings here and the results are not manipulated by side effects of different settings.

Figure 4.1 contrasts the confusion matrix of modeling I with the one of modeling II. In modeling I (Figure 4.1a), the highest values are arranged along the diagonal, which means that most of the predictions agree with the actual target class (ground truth).

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**4**	0	0	0	0	0	0	1	2	0	0	0	0	0
bowl.n.01	0	**2**	0	0	0	0	0	0	0	0	0	1	0	0
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**4**	0	0	0	0	1	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	2	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	2	**0**	0	0	0	0	0	0	2
lemon.n.01	0	0	0	0	0	0	0	**2**	2	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	1	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**3**	0	1	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**4**	0	0	0
pot.n.01	0	2	0	0	0	0	0	0	0	0	0	**2**	0	0
spatula.n.02	0	0	0	0	0	0	2	0	0	0	0	0	**4**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**

(a) Confusion matrix for modeling I and configuration I of the formula template

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**4**	0	0	0	0	0	0	1	2	0	0	0	0	0
bowl.n.01	0	**2**	0	0	3	2	1	0	0	2	2	2	0	2
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	2	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	1	0	0	0	0	0	0	1
knife.n.01	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**2**	2	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	1	**0**	0	0	0	0	0
pan.n.01	0	1	0	0	1	0	0	0	0	**2**	0	1	0	1
plate.n.04	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0
pot.n.01	0	1	0	0	0	0	0	0	0	0	0	**1**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

(b) Confusion matrix for modeling II and configuration I of the formula template

Figure 4.1: Comparing the two different modelings from Section 2.5

The matrix indicates that the worst results in modeling I are achieved for knives and oranges and in modeling II for oranges, cups and spoons, as none of them was predicted correctly. The respective predictions, however, voted for similar concepts with similar descriptions.

The outliers therefore mostly indicate the confusion of semantically similar classes like pot and bowl, *lemon* and *orange* or spoon and knife. These classes have similar descriptions, hence similar property attributes.

Pots and bowls for example are both subordinate concepts of containers and may also be similar in shape and size.

Spoon and knife, similarly, are both objects categorized as cutlery and have handles (see Listings 4.1 and 4.2).

```
hypernym(cutlery.n.02)
hasa(handle.n.01)
hasa(blade.n.09)
```

Listing 4.1: Example Descriptions for *knife.n.01*

```
hypernym(cutlery.n.02)
size(shallow.a.01)
shape(bowl-shaped.s.01)
hasa(handle.n.01)
```

Listing 4.2: Example Descriptions for *spoon.n.01*

Other objects that are easily confused are the different types of fruit, especially the ones with similar colors (e.g. an orange, which is described as a *round, orange fruit* and a lemon which is a *yellow, oval fruit*).

The confusion matrix for modeling II (Figure 4.1b) shows more confusions, also between different object classes. It shows a strong preference for bowls, which can be seen in the second row. Even fruits (orange) and very dissimilar objects like forks and knives are falsely classified as bowls. An explanation for this maybe the overlapping attribute domains.

In modeling II, it is not differentiated between property types when grounding the MLN, which usually leads to formulas that do not make sense in the real world, as property types for certain values are used interchangably.

In some cases, however, formulas with a *different* meaning are generated, instead of producing a non-expressive formula. This may be the case when confusing part-of attributes with hypernyms. As an example, the training examples contain a description, according to which a spoon *has a* bowl, resulting from the NL description "A spoon is a utensil consisting of a small shallow bowl[...]", while other objects (like the bowl itself, a cup or a pot) are usually described as subconcepts of container, which is closely related to bowl. Mixing up the *HASA* and *HYPERNYM* relations may lead to different formula weights and therefore different decisions in the inference process.

The observations from the confusion matrices are underpinned by the applied quality measures. The results for the MLN generated from the configuration I template yield a better F1 score for modeling I (0.64), than for modeling II (0.46).

The recall values for pots, bowls and spoons are 0.5, for knives even 0.0 and also low precisions of 0.0 to 0.67 (except for 1.0 for spoon). This explains the low F1 scores, as it is the harmonic mean of these two measures.

The best results (an F1 score of 1.0) are achieved for plates and cherries in modeling I and coffee in both modelings. The worst results that can be observed - as already indicated by the confusion matrix - are the ones for knives and oranges in modeling I and oranges, cups and spoons in modeling II.

4.2.2 COMPARING RESULTS UNDER OPEN AND CLOSED WORLD ASSUMPTIONS

Previously, the two different modelings were compared using configuration I as an example. Now the impact of an open world assumption on the overall result is investigated. We use the same modeling this time and compare the results with and without the closed world assumption. We can not use modeling II here because, as already mentioned before, a closed world assumption is made implicitly in any case. We therefore choose modeling I and again, opt for configuration I. The resulting confusion matrices can be seen in Figure 4.2.

Here, the discrepancy is even more evident than in the example before, as almost no class has been correctly classified, looking at Figure 4.2a, which shows the results under the open world assumption.

To approach an explanation for this perfomance, we need to recall the setting of the formula templates. In configuration I, each formula consists of at most 3 atoms, of which one is the query object. In other words, one formula defines an object by only two attributes in most cases. The total of true formulas given the evidence then determines the object.

Gradually assuming a certain atom to be true (=attribute to be present) and retrieving the number of formulas that would be evaluated to true in that case, leads to a loss of expressiveness, as almost every configuration of one evident and one assumed property is possible. The combination of several such formulas identify different objects to be equally probable or predict a wrong object to be most probable. Unsurprisingly, a strong preference for objects with detailed descriptions (i.e. many property attributes in the training set) like spoons, pots and bananas is noticable, as more formulas vote for these objects.

The confusion matrix in Figure 4.2b shows the results for the same configuration but under a closed world assumption. Here, only the actually perceived evidence is assumed to be true, everything not present in the evidence database is therefore considered false. Usually, natural-language descriptions of a certain object do not differ much from each other, which is why the training data often contains identical or semantically similar property attributes.

This similarity is used in the inference step when adding similarity information. The effect of this practice is that an object with the properties *orange* and *oval* can still match a lemon's description “yellow and oval”, as the information about the similarity between *orange* and yellow is added to the evidence. Therefore, an *orange* and oval object is not entirely unlikely to be a lemon, but only less probable than an object matching the description perfectly.

The performance measures show the significance in the difference between the two compared concepts. Without the closed world assumption, the highest F1 score for the overall average value is only 0.05 and 0.4 at most for single objects, in this case bowls. All other single objects only reach values lower or equal to 0.13. Low values can therefore also be observed for precision and recall. Only one object has a high

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**0**	0	1	0	4	0	2	0	2	1	0	2	1	2
bowl.n.01	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
cherry.n.03	0	0	**0**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**0**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	0	0	0	0	**0**	0	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0
pot.n.01	1	2	0	2	0	0	0	2	1	0	0	**1**	2	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0
spoon.n.01	3	1	3	2	0	4	2	2	1	3	4	1	1	**2**

(a) Confusion matrix for modeling I and configuration I of the formula template (open world assumption).

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**4**	0	0	0	0	0	0	1	2	0	0	0	0	0
bowl.n.01	0	**2**	0	0	0	0	0	0	0	0	0	1	0	0
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**4**	0	0	0	0	1	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	2	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	2	**0**	0	0	0	0	0	0	2
lemon.n.01	0	0	0	0	0	0	0	**2**	2	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	1	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**3**	0	1	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**4**	0	0	0
pot.n.01	0	2	0	0	0	0	0	0	0	0	0	**2**	0	0
spatula.n.02	0	0	0	0	0	0	2	0	0	0	0	0	**4**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**

(b) Confusion matrix for modeling I and configuration I of the formula template (closed world assumption).

Figure 4.2: Comparing the inference results with and without a closed world assumption

precision of 1.0 (bowl), the next highest is 0.09 for pot. Most of the objects reached a precision value under 0.1. and a recall value lower or equal to 0.25.

When making a closed world assumption the results are much better, as an F1 score of 0.63 is reached on average and 1.0 for several single objects (plates, cherries, coffee).

4.2.3 COMPARING CONFIGURATIONS

From the previous sections it can be seen that modeling I is the one to go for and that a closed world assumption is to be preferred over an open world assumption. What has not been examined yet, is the best configuration of formula templates to generate a model that represents our object-property relations best. Different configurations have been tested, of which we present three here. Each of the configurations can be found in the appendix (see MLN Templates - Modeling I), but the difference is outlined shortly.

The first configuration (configuration I) consists of 5 formula templates, each a conjunction of one atom for object identity (which is the one we query for) and one or two atoms for the property attributes. Three of the formulas are simply all 2-pair combinations of the properties size, shape and color, the remaining two only contain one atom for the part-of and hypernym attribute, respectively.

Configuration II extends configuration I by two formulas combining colors and hypernyms and shapes and hypernyms, respectively. These combinations are based on observations of how object descriptions usually look like. Describing objects in natural language, humans tend to name the superordinate concept in combination with a descriptive attribute, such as color or shape, which is why these formulas were designed this way.

Configuration III consists of one larger formula, comprising the three visual attributes size, shape and color and two additional formulas only containing atoms for part-of and hypernym attributes. This configuration is designed to represent more complex object descriptions and is therefore more restrictive than the other ones. Figure 4.3 shows the confusion matrices for the respective configurations, each using modeling I and making a closed world assumption.

At first view, the three configurations (especially I and II) seem to only slightly differ in their results, but taking a closer look, configuration II turns out to generate a model that fits the data best. Configuration III fits the data least and gives overall poor results compared to the other two configurations.

As a reminder, configuration III consists of only three formula templates. One formula is a conjunction of the predicates *object, color, size* and *shape*, the other two are conjunctions of an *object* atom and a *hypernym* or *hasa* atom, respectively. Most of the formulas generated from the templates have a weight of 0, as none of the training examples contains attributes for color, size AND shapes at the same time. It is simply not very common to describe an object using more than two types of attributes. Interestingly, the results are still not as bad as would be expected. The reason is that the inference based on this model only relies on the hypernym and part-of attributes of the objects, which seem to be rather discriminative in the underlying dataset.

As an example, the cherry is not confused with any other object, as it *hasa* stone, which is not true for any other object. The coffee is a subordinate concept (*hypernym*) of liquid or beverage, which is also a unique characteristic in the dataset.

Data generated from ROBOSHERLOCK would suit this configuration better as they are generated from the annotations of the perception of example objects. Such annotations *always* contain information about size, shape and color of an object, next to others like texture or logos. On the other hand, they do not (yet) provide information about *hypernym* or *hasa* relations, which might worsen the result.

This approach rather concentrates on recognizing certain instances of an object, learning from training examples and using a model to identify similar objects. It only generalizes to a certain extent, limited by what it has learned from the examples.

The configurations I and II are better suited for data generated from natural-language descriptions as they incorporate the form in which objects are described naturally. Therefore they are designed to create a model representing a general concept of an object, rather than representing a description of specific instances.

As mentioned above, the overall performance of configuration I and II only slightly differ from each other comparing their average values (see Table 4.2). Still, configuration II seems to be the better option, as it shows that incorporating background knowledge about frequent property attribute combinations in common natural-language descriptions pays off as it tends to achieve better results.

4.2.4 INCORPORATING SIMILARITY

Taking a look at Figure 4.1, 4.2 and 4.3 and their respective quality measures gives us an impression about the overall performance of the MLNs used. The results presented above are calculated using the classic quality measures *accuracy*, *precision*, *recall* and *F1 score* as mentioned before. The problem is that these measures do not take the semantic similarity of the target classes into account. In other words, two objects with similar descriptions might be classified interchangably and therefore rated as erroneous, which is not a problem of the modeling, but the underlying training data. In particular, a decision of the model may be "correct" according to the dataset, even though it did not predict the expected class. This is due to the fact that not all of the objects are described discriminatively.

As an example, the two objects *orange* and *lemon* have more property attributes in common than distinguishing them according to the dataset. Both are subordinate classes of fruit and both have colors (yellow and orange) and shapes (oval and round) which in turn are very similar. These slight differences in their descriptions are not discriminative for the two fruits and therefore they are easily confused.

If the data do not sufficiently represent the differences of the object, the results of the quality measures do not necessarily represent the overall quality of the trained model, but rather the quality of the object descriptions, i.e. the underlying training data. From the confusion matrices issued above can be learned that similar objects such as *orange* and *lemon* or fork and knife, are more likely to be confused than very differently described ones, such as pan and banana. This is not a problem of

the model but of the underlying data. An indefinite description like "A round yellow fruit" is most likely to cause a human to confuse oranges and lemons as well.

One could also say that the error of confusing similar objects such as a *lemon* and an ORANGE is less profound than the error of confusing totally different objects like *orange* and CUP. In a kitchen environment, it would be acceptable if the robot chose to take a knife to stir the batter instead of a spoon, but using an *orange* would definitely be considered inappropriate.

To show, how the similarity of objects influences the error rate, the quality measures have been altered in a way that the calculation takes into account, how much the prediction differs from the actual class. That means that each false positive example (and false negative, respectively) is weighted with *1-similarity(p, t)*, *p* being the predicted class and *t* the actual class (ground truth) of the arguable object. The calculations for the respective measures itself are left untouched.

Taking the performance of the MLN generated from modeling I and configuration II (using a closed world assumption) as an example, one can already see from Figure 4.4 how taking the object similarities into account influences the overall result. Especially the values in the center of the matrix, indicating the confusion of forks and knives, show a significant difference between the conventional error calculation and the calculation including the similarity (4 vs. 1.74 and 2 vs. 0.87, respectively).

The overall quality measures have changed considerably as well. While the F1-score returns a value of 0.67 using the classic calculation, adding the similarity weighting increases it by 0.14 to 0.81. In all modelings and configurations, all four modified quality measures return better results, which is obvious, given the fact that each incorrect classification is contributing less to the overall error than before, while the correct classifications are contributing most (as they are weighed with 1.0, due to the identity relation between predicted class and ground truth).

The difference in the results from classic and modified calculation can be seen in Table 4.2.

Acc	Prec	Rec	F1
0.95	0.72	0.68	0.67
0.98	0.83	0.80	0.81
3.16%	**15.28%**	**17.64%**	**20.89%**

Table 4.2: Results - Similarity: comparing the results of the configuration using classic measures and measures incorporating the similarity between concepts. The last row (printed in boldface) is the procentual increase achieved by incorporating the similarity.

Incorporating the similarity results in a 14.24% increase on average over the four individual average performance measures in this setting, which is not to be confused with a 14.24% better result on the data.

4.3 WHICH ONE IS THE BEST? - SUMMARY

According to our findings in the previous sections, we conclude that the MLN generated using modeling I and configuration II (under a closed world assumption) works best to represent the underlying data. Table 4.3 shows the overall results of all tested settings, of which the best ones are highlighted in green. Except for two equivalent accuracy values in configuration I, our chosen setting outperforms the others in each performance category.

		A	**A**$_{sim}$	**P**	**P**$_{sim}$	**R**	**R**$_{sim}$	**F1**	**F1**$_{sim}$
Modeling I	CONF I	0.87	0.93	0.08	0.09	0.07	0.12	0.05	0.09
	CONF I-cw	0.95	0.98	0.64	0.77	0.66	0.77	0.64	0.77
	CONF II	0.87	0.92	0.05	0.07	0.07	0.15	0.04	0.08
	CONF II-cw	0.95	0.98	0.72	0.83	0.67	0.81	0.67	0.81
	CONF III	0.86	0.91	0.09	0.10	0.09	0.14	0.05	0.09
	CONF III-cw	0.94	0.98	0.53	0.60	0.57	0.62	0.54	0.61
Modeling II	CONF I	0.92	0.97	0.51	0.64	0.48	0.61	0.46	0.60
	CONF I	0.92	0.97	0.52	0.61	0.48	0.59	0.47	0.56
	CONF III	0.91	0.96	0.49	0.59	0.39	0.52	0.40	0.53

Table 4.3: Average quality measures for all modelings and configurations with and without closed world assumption: Accuracy (A), Precision (P), Recall (R) and F1-Score (F1) with and without similarity consideration ($_{sim}$), respectively. The best results are highlighted in green.

The conclusions we draw from the evaluation can be summed up as follows:

- The modeling using separate domains for different feature types instead of a shared one are to be favored, as it is less complex in the generation process and provides better results at the same time.
- A closed world assumption is to be preferred over an open world assumption.
- Taking knowledge about the typical structure of natural-language descriptions during the design process of the MLN formula templates benefits the performance of the generated model.
- Considering the respective similarities of the target classes when measuring the performance helps to get a better overview over the quality of the *model* instead of the quality of the underlying *data*.

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**4**	0	0	0	0	0	0	1	2	0	0	0	0	0
bowl.n.01	0	**2**	0	0	0	0	0	0	0	0	0	1	0	0
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**4**	0	0	0	0	1	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	2	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	2	**0**	0	0	0	0	0	0	2
lemon.n.01	0	0	0	0	0	0	0	**2**	2	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	1	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**3**	0	1	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**4**	0	0	0
pot.n.01	0	2	0	0	0	0	0	0	0	0	0	**2**	0	0
spatula.n.02	0	0	0	0	0	0	2	0	0	0	0	0	**4**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**

(a) Confusion matrix for modeling I and configuration I of the formula template

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**4**	0	0	0	0	0	0	1	2	0	0	0	0	0
bowl.n.01	0	**2**	0	0	0	0	0	0	0	0	0	2	0	0
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**4**	0	0	0	0	2	0	0	0	0
fork.n.01	0	0	0	0	0	**0**	2	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	4	**2**	0	0	0	0	0	0	2
lemon.n.01	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	1	**2**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0
plate.n.04	0	1	0	0	0	0	0	0	0	0	**4**	0	0	0
pot.n.01	0	1	0	0	0	0	0	0	0	0	0	**2**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**4**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**

(b) Confusion matrix for modeling I and configuration II of the formula template

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**0**	0	0	0	0	0	0	2	0	0	0	0	0	0
bowl.n.01	0	**0**	0	0	0	0	0	0	0	0	0	2	0	0
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**4**	0	0	0	0	1	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	2	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	2	**2**	0	0	0	0	0	0	1
lemon.n.01	3	0	0	0	0	0	0	**0**	0	0	0	0	0	0
orange.n.01	1	0	0	0	0	0	0	2	**4**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**3**	0	1	0	0
plate.n.04	0	2	0	0	0	0	0	0	0	0	**2**	1	0	0
pot.n.01	0	2	0	0	0	0	0	0	0	0	2	**0**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**4**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**

(c) Confusion matrix for modeling I and configuration III of the formula template

Figure 4.3: Comparing the inference results for the three different formula template configurations.

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**4**	0	0	0	0	0	0	1	2	0	0	0	0	0
bowl.n.01	0	**2**	0	0	0	0	0	0	0	0	0	2	0	0
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**4**	0	0	0	0	2	0	0	0	0
fork.n.01	0	0	0	0	0	**0**	2	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	4	**2**	0	0	0	0	0	0	2
lemon.n.01	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	1	**2**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0
plate.n.04	0	1	0	0	0	0	0	0	0	0	**4**	0	0	0
pot.n.01	0	1	0	0	0	0	0	0	0	0	0	**2**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**4**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**

(a) Confusion matrix for modeling I and configuration I of the formula template (classic quality measures).

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**4**	0	0	0	0	0	0	0.78	1.57	0	0	0	0	0
bowl.n.01	0	**2**	0	0	0	0	0	0	0	0	0	1.68	0	0
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**4**	0	0	0	0	1.2	0	0	0	0
fork.n.01	0	0	0	0	0	**0**	0.87	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	1.74	**2**	0	0	0	0	0	0	1.14
lemon.n.01	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	0.75	**2**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0
plate.n.04	0	0.5	0	0	0	0	0	0	0	0	**4**	0	0	0
pot.n.01	0	0.84	0	0	0	0	0	0	0	0	0	**2**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**4**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**

(b) Confusion matrix for modeling I and configuration I of the formula template (quality measures considering similarity).

Figure 4.4: Comparing the inference results under consideration of similarity.

CHAPTER five

SUMMARY

This chapter summarizes the results of the preceding evaluation and draws a conclusion from the results. Furthermore, future prospects are proposed.

5.1 CONCLUSIONS

We investigated the question, if it is possible to recognize objects only from natural-language descriptions. We have investigated the problem following from this question and identified the subordinate problems that were to solve, to be able to answer that question.

In this thesis, we proposed a method to transfer natural-language descriptions into a formal, first-order logic based representation, to be able to use them as training examples for a probabilistic relational model. This model has then been evaluated using a 10-fold cross-validation, which has shown that it is possible to recognize objects only based on their symbolic features. In order to define similarity relations between symbolic features and therefore render a comparison of object descriptions possible, we proposed three similarity calculations for the feature types, either based on the respective characteristics of the features or their taxonomic relatedness.

According to the evaluation results, the most suitable model design is based on observations of how object descriptions usually look like and is to be preferred over long formulas, taking numerous feature types into account.

During the training period, it became clear that the quality of the model highly correlates with the quality of the similarity specification. Using the provided (mostly path-based) similarity functions from WORDNET may result in a very unintuitive

similarity measure (according to the original WUP-similarity, the according synsets for the semantically rather dissimilar words fruit and container have a comparably high similarity of 0.5, which complicates the distinction between fruits and other objects).

Another problem is the use of relative features in object descriptions. Object descriptions from the internet do not follow any rules, which means that many descriptions may be entirely useless for object recognition tasks. Terms like "larger than" or "smaller than" are challenging, as they do not refer to a specific, measurable value, but rather compare attributes of two objects directly. These terms can not be used synonymously with "large" or "small", as for example, a melon may be large compared to a cherry, but small compared to a car. Similarly, objects are often described by comparing it to another similar object. Therefore the descriptions rather contain the differences between the two objects than a purely object-related specification and do not mention (possibly too obvious) attributes, which might have been useful and easy to recognize.

Problematic is also the use of features that are not recognizable by the robot like certain *hasa* relations (seeds, stone) or features that are not discriminative (similar objects are described similarly, for example cutlery).

In general, object descriptions from the internet require preprocessing steps to filter irrelevant information and ideally, to put them into a syntactically correct form.

The generated model can be used with actual data from ROBOSHERLOCK, but as the annotations currently differ from the feature types in the model, the results are poor at present. In particular, ROBOSHERLOCK provides information about text or logos that have been found on the objects, which is usually not found in natural language object descriptions. On the other hand, our model comprises features denoting hypernym and part-of relations, which is not or only partly implemented in ROBOSHERLOCK.

Furthermore, the perception pipeline always outputs the perceived features of the selected annotators, but does not differentiate in the significance of the features. That means that if a color annotator is chosen, it will output the colors of the perceived object, even if it is not significant for the object. This way of describing objects does not necessarily match the way, a human would describe it, which is what our model is based on.

To conclude the findings described above and to answer the initial question, it is possible to recognize objects based on natural-language descriptions, if an appropriate similarity is defined between object descriptions and the perception pipeline is able to output annotations that match the features represented by the respective model.

5.2 PERSPECTIVE

We propose that incorporating additional features into the model as a subject for future investigations. Object descriptions in natural language comprises numerous ways to characterize objects, which is in general not limited to the 5 feature types used in this work. In particular, potentially perceivable attributes such as materials or surfaces might increase the performance of a trained model, especially if they represent potentially discriminative characteristics of the objects.

Aldoma et al. [1] and Varadarajan and Vincze ([33] and [34]) identify semantic affordance features for grasping previously unseen objects and use part detection based on semantic cues. These affordances can as well be interpreted as attributes, for example the affordance "contain-ability" matches the purpose "holding liquids". NL object descriptions often contain information about an application purpose of an object, i.e. what it can be used for and how. This information can be interpreted as a used-for attribute as well and matched to the affordances that have been determined for a perceived object. More relations between the affordances of an object and its application purposes according to its NL description are imaginable and are worth further investigations.

As the MLN represents a joint distribution over observations, it is possible to reason about the most significant attributes for certain objects. This knowledge can be used in the design process of future models for object-attribute relations.

We proposed to use for object recognition and introduced an approach how this can be put into execution using Markov Logic Networks. Our findings show that this approach signifies one step further towards object recognition tasks incorporating knowledge-based features.

Acronyms

EM Expectation Maximization. 8

FOL first-order logic. 17, 21

HOG Histogram of Oriented Gradients. 3

HSV Hue Value Saturation. 3, 5, 34, 35, 88

LCS Least Common Subsumer. 33, 34, 39, 40

MAP Maximum a Posteriori. 19

MLN Markov Logic Network. 17–23, 25–29, 31, 32, 41, 44, 45, 51–54, 57, 59, 61, 62, 64, 68–70, 75

MPE most probable explanation. 20, 52, 53

MRF Markov Random Field. 17, 18

NL Natural Language. 1, 5–8, 14, 15, 21, 23, 27, 45, 52, 53, 64, 75

NLP Natural Language Processing. 8

PCFG probabilistic context-free grammars. 23

POS part-of-speech. 23, 24, 26, 28, 32, 33

PRAC Probabilistic Robot Action Cores. 51

PRM probabilistic relational model. 17, 27

RANSAC Random sample consensus. 5

SIFT Scale-invariant feature transform. 3

UIM Unstructured Information Management. 4

WCSP weighted constraint satisfaction problem. 19, 20, 52, 53

REFERENCES

[1] A. Aldoma, F. Tombari, and M. Vincze. Supervised learning of hidden and non-hidden 0-order affordances and detection in real scenes. In *2012 IEEE International Conference on Robotics and Automation*, pages 1732–1739. IEEE, May 2012. ISBN 978-1-4673-1405-3.

[2] P. Chang and J. Krumm. Object recognition with color cooccurrence histograms. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[3] H.-T. Cheng, F.-T. Sun, M. Griss, P. Davis, J. Li, and D. You. Nuactiv: Recognizing unseen new activities using semantic attribute-based learning. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 361–374. ACM, 2013.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005.

[5] K. Duan, D. Parikh, D. Crandall, and K. Grauman. Discovering Localized Attributes for Fine-grained Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[6] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing Objects by their Attributes. In *CVPR*, 2009.

[7] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar. Probabilistic Relational Models. In *Introduction to Statistical Relational Learning*, chapter 5, pages 129–174. MIT Press, 2007.

[8] D. Gross and K. J. Miller. Adjectives in wordnet. *International Journal of Lexicography*, 3(4):265–277, 1990. URL `http://ijl.oxfordjournals.org/content/3/4/265.abstract`.

[9] S. Guadarrama, E. Rodner, K. Saenko, N. Zhang, R. Farrell, J. Donahue, and T. Darrell. Open-vocabulary object retrieval. RSS, 2014.

[10] D. Jain. *Probabilistic Cognition for Technical Systems: Statistical Relational Models for High-Level Knowledge Representation, Learning and Reasoning*. PhD thesis, München, Technische Universität München, Diss., 2012.

[11] D. Jain, P. Maier, and G. Wylezich. Markov logic as a modelling language for weighted constraint satisfaction problems. *Constraint Modelling and Reformulation (ModRef'09)*, page 60, 2009.

[12] D. Jayaraman, F. Sha, and K. Grauman. Decorrelating semantic visual attributes by resisting the urge to share. In *CVPR*, 2014.

[13] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[14] S. Kok. *Structure Learning in Markov Logic Networks*. PhD thesis, University of Washington, 2010.

[15] C. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(3):453–465, March 2014. ISSN 0162-8828.

[16] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.

[17] S. Lauriar, G. Bugmann, T. Kyriacou, J. Bos, and E. Klein. Training Personal Robots Using Natural Language Instruction. *IEEE Intelligent Systems*, 16(5): 38–45, 2001. ISSN 1541-1672.

[18] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[19] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. ISSN 0920-5691. URL `http://dx.olddoi.org/10.1023/B%3AVISI.0000029664.99615.94`.

[20] C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. A Joint Model of Language and Perception for Grounded Attribute Learning. *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, June 2012.

[21] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. Learning to Parse Natural Language Commands to a Robot Control System. In V. Desai, Jaydev P. and Dudek, Gregory and Khatib, Oussama and Kumar, editor, *Experimental Robotics*, pages 403–415. Springer International Publishing, 88 edition, 2013. ISBN 978-3-319-00064-0.

[22] D. Nyga and M. Beetz. Everything robots always wanted to know about housework (but were afraid to ask). *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 243–250, Oct 2012.

[23] D. Nyga and M. Beetz. Incorporating class taxonomies in probabilistic relational models. (under review), 2014.

[24] D. Nyga, F. Balint-benczedi, and M. Beetz. PR2 Looking at Things – Ensemble Learning for Unstructured Information Processing with Markov Logic Networks.

[25] M. Palmer and Z. Wu. VERB SEMANTICS AND LEXICAL SELECTION. *ACL '94 Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138, 1994.

[26] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, Jan. 2006. ISSN 0885-6125.

[27] P. Singla and P. Domingos. Discriminative Training of Markov Logic Networks. *AAAI'05 Proceedings of the 20th national conference on Artificial intelligence*, 2: 868–873, 2005.

[28] P. Singla and P. Domingos. Entity Resolution with Markov Logic. *Sixth International Conference on Data Mining (ICDM'06)*, pages 572–582, Dec. 2006. ISSN 1550-4786.

[29] P. Singla and P. Domingos. Markov logic in infinite domains. *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI2007)*, pages 368–375, 2007.

[30] P. Singla and P. Domingos. Lifted first-order belief propagation. In *AAAI*, volume 8, pages 1094–1099, 2008.

[31] Y. Su, M. Allan, and F. Jurie. Improving object classification using semantic attributes. In *BMVC*, pages 1–10, 2010.

[32] M. Tenorth, D. Nyga, and M. Beetz. Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1486–1491, Anchorage, AK, USA, May 3–8 2010.

[33] K. M. Varadarajan and M. Vincze. Object part segmentation and classification in range images for grasping. *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 21–27, June 2011.

[34] K. M. Varadarajan and M. Vincze. AfRob: The affordance network ontology for robots. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1343–1350, Oct. 2012.

[35] F. X. Yu, L. Cao, R. S. Feris, J. R. Smith, and S.-F. Chang. Designing category-level attributes for discriminative visual recognition. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 771–778. IEEE, 2013.

APPENDIX

MLN TEMPLATES - MODELING I

```
// predicates
shape(cluster, shape)
color(cluster, color)
size(cluster, size)
hypernym(cluster, hypernym)
hasa(cluster, hasa)
object(cluster, object!)

// formula templates

// CONFIGURATION I
0 object(?c, +?objID) ^ size(?c, +?size) ^ shape(?c, +?shape)
0 object(?c, +?objID) ^ color(?c, +?color) ^ shape(?c, +?shape)
0 object(?c, +?objID) ^ color(?c, +?color) ^ size(?c, +?size)
0 object(?c, +?objID) ^ hasa(?c, +?hasa)
0 object(?c, +?objID) ^ hypernym(?c, +?hyp)


// CONFIGURATION II
0 object(?c, +?objID) ^ size(?c, +?size) ^ shape(?c, +?shape)
0 object(?c, +?objID) ^ color(?c, +?color) ^ shape(?c, +?shape)
0 object(?c, +?objID) ^ color(?c, +?color) ^ size(?c, +?size)
0 object(?c, +?objID) ^ color(?c, +?color) ^ hypernym(?c, +?hyp)
0 object(?c, +?objID) ^ shape(?c, +?shape) ^ hypernym(?c, +?hyp)
0 object(?c, +?objID) ^ size(?c, +?size) ^ hypernym(?c, +?hyp)
0 object(?c, +?objID) ^ hasa(?c, +?hasa) ^ hypernym(?c, +?hyp)
0 object(?c, +?objID) ^ hasa(?c, +?hasa)
0 object(?c, +?objID) ^ hypernym(?c, +?hyp)


// CONFIGURATION III
0 object(?c, +?objID) ^ color(?c, +?col) ^ size(?c, +?size)
                      ^ shape(?c, +?shp)
0 object(?c, +?objID) ^ hasa(?c, +?hasa)
0 object(?c, +?objID) ^ hypernym(?c, +?hyp)
```

Listing 5.1: MLN Templates - Modeling I

MLN TEMPLATES - MODELING II

```
// predicates
prop(cluster, word, prop)
object(cluster, object!)

// formula templates

// CONFIGURATION I
0 object(?c, +?objID) ^ prop(?c, +?size, SIZE) ^ prop(?c, +?shape, SHAPE)
0 object(?c, +?objID) ^ prop(?c, +?color, COLOR) ^ prop(?c, +?shape, SHAPE)
0 object(?c, +?objID) ^ prop(?c, +?color, COLOR) ^ prop(?c, +?size, SIZE)
0 object(?c, +?objID) ^ prop(?c, +?hasa, HASA)
0 object(?c, +?objID) ^ prop(?c, +?hyp, HYPERNYM)


// CONFIGURATION II
0 object(?c, +?objID) ^ prop(?c, +?size, SIZE) ^ prop(?c, +?shape, SHAPE)
0 object(?c, +?objID) ^ prop(?c, +?color, COLOR) ^ prop(?c, +?shape, SHAPE)
0 object(?c, +?objID) ^ prop(?c, +?color, COLOR) ^ prop(?c, +?size, SIZE)
0 object(?c, +?objID) ^ prop(?c, +?color, COLOR) ^ prop(?c, +?hyp, HYPERNYM
    )
0 object(?c, +?objID) ^ prop(?c, +?shape, SHAPE) ^ prop(?c, +?hyp, HYPERNYM
    )
0 object(?c, +?objID) ^ prop(?c, +?size, SIZE) ^ prop(?c, +?hyp, HYPERNYM)
0 object(?c, +?objID) ^ prop(?c, +?hasa, HASA) ^ prop(?c, +?hyp, HYPERNYM)
0 object(?c, +?objID) ^ prop(?c, +?hasa, HASA)
0 object(?c, +?objID) ^ prop(?c, +?hyp, HYPERNYM)


// CONFIGURATION III
0 object(?c, +?objID) ^ prop(?c, +?col, COLOR) ^ prop(?c, +?size, SIZE)
                    ^ prop(?c, +?shp, SHAPE)
0 object(?c, +?objID) ^ prop(?c, +?hasa, HASA)
0 object(?c, +?objID) ^ prop(?c, +?hyp, HYPERNYM)
```

Listing 5.2: MLN Templates - Modeling II

MLN Templates - Property attributes

```
// include predicates used by the stanford parser (syntactic evidence)
#include ../../nl_parsing/mln/predicates.mln

has_sense(word, sense!)
is_a(sense, concept)
property(word, prop)

0 nsubj(?w1, ?w2) ^ has_pos(?w1, JJ) ^ has_sense(?w1, ?s1) ^ property(?w1,
    +?prop) ^ ?w1 =/= ?w2

0 amod(?w1,?w2) ^ has_pos(?w2, JJ) ^ has_sense(?w2, ?s2) ^ is_a(?s2, +?c) ^
     property(?w2, +?prop) ^ ?w1 =/= ?w2

0 dobj(?w1, ?w2) ^ has_sense(?w1, ?s1) ^ is_a(?s1, own.v.01) ^ has_sense(?
    w2, ?s2) ^ property(?w2, HASA) ^ ?w1 =/= ?w2
0 dobj(?w1, ?w2) ^ has_sense(?w1, ?s1) ^ is_a(?s1, own.v.01) ^ has_sense(?
    w2, ?s2) ^ !property(?w2, HASA) ^ ?w1 =/= ?w2
0 dobj(?w1, ?w2) ^ has_sense(?w1, ?s1) ^ is_a(?s1, envelop.v.01) ^
    has_sense(?w2, ?s2) ^ property(?w2, HASA) ^ ?w1 =/= ?w2
0 dobj(?w1, ?w2) ^ has_sense(?w1, ?s1) ^ is_a(?s1, envelop.v.01) ^
    has_sense(?w2, ?s2) ^ !property(?w2, HASA) ^ ?w1 =/= ?w2

0 cop(?w1,?w2) ^ has_pos(?w1, NN) ^ has_sense(?w1,?s1) ^ is_a(?s1, +?c) ^
    property(?w1, HYPERNYM) ^ ?w1 =/= ?w2
0 cop(?w1,?w2) ^ has_pos(?w1, NN) ^ has_sense(?w1,?s1) ^ is_a(?s1, +?c) ^ !
    property(?w1, HYPERNYM) ^ ?w1 =/= ?w2
0 cop(?w1,?w2) ^ has_pos(?w1, JJ) ^ has_sense(?w1,?s1) ^ is_a(?s1, +?c) ^
    property(?w1, +?prop) ^ ?w1 =/= ?w2

0 conj_or(?w1, ?w2) ^ has_pos(?w1,JJ) ^ property(?w1, +?prop) ^ ?w1 =/= ?w2
0 conj_or(?w1, ?w2) ^ has_pos(?w2,JJ) ^ property(?w2, +?prop) ^ ?w1 =/= ?w2

0 conj_and(?w1, ?w2) ^ has_pos(?w1,JJ) ^ property(?w1, +?prop) ^ ?w1 =/= ?
    w2
0 conj_and(?w1, ?w2) ^ has_pos(?w2,JJ) ^ property(?w2, +?prop) ^ ?w1 =/= ?
    w2

0 prep_with(?w1,?w2) ^ has_sense(?w2,?s2) ^ property(?w2, HASA) ^ ?w1 =/= ?
    w2
0 prep_with(?w1,?w2) ^ has_sense(?w2,?s2) ^ !property(?w2, HASA) ^ ?w1 =/=
    ?w2
0 prep_without(?w1,?w2) ^ property(?w2, HASA) ^ ?w1 =/= ?w2
0 prep_without(?w1,?w2) ^ !property(?w2, HASA) ^ ?w1 =/= ?w2

// prevent verbs from being detected as hypernyms
0 has_pos(?w1, +?pos) ^ property(?w1, +?prop)
```

Listing 5.3: MLN Templates - Property attributes

Syntactic Evidence Example

```
// spoon.n.01 / spoon%1:06:00::
// It is a piece of cutlery with a shallow bowl-shaped container and a
    handle.
// It is used to stir or serve or take up food

// STANFORD:
amod(container-11,bowl-shaped-10)
amod(container-11,shallow-9)
aux(stir-20,to-19)
auxpass(used-18,is-17)
conj_and(container-11,handle-14)
conj_or(stir-20,serve-22)
conj_or(stir-20,take-24)
cop(piece-4,is-2)
det(container-11,a-8)
det(handle-14,a-13)
det(piece-4,a-3)
dobj(take-24,food-26)
has_pos(It-1,PRP)
has_pos(It-16,PRP)
has_pos(a-13,DT)
has_pos(a-3,DT)
has_pos(a-8,DT)
has_pos(bowl-shaped-10,JJ)
has_pos(container-11,NN)
has_pos(cutlery-6,NN)
has_pos(food-26,NN)
has_pos(handle-14,NN)
has_pos(is-17,VBZ)
has_pos(is-2,VBZ)
has_pos(piece-4,NN)
has_pos(serve-22,VB)
has_pos(shallow-9,JJ)
has_pos(stir-20,VB)
has_pos(take-24,VB)
has_pos(to-19,TO)
has_pos(up-25,RP)
has_pos(used-18,VBN)
nsubj(piece-4,It-1)
nsubjpass(used-18,It-16)
prep_of(piece-4,cutlery-6)
prep_with(cutlery-6,container-11)
prt(take-24,up-25)
rcmod(container-11,used-18)
root(ROOT-0,piece-4)
xcomp(used-18,stir-20)
```

Listing 5.4: Syntactic Evidence Example

```
// WORDNET ANNOTATED:
has_sense(It-1,null)
has_sense(is-2,null)
has_sense(a-3,null)
has_sense(piece-4,null)
has_sense(cutlery-6,cutlery.n.02)
has_sense(a-8,null)
has_sense(shallow-9,shallow.a.01)
has_sense(bowl-shaped-10,bowl-shaped.s.01)
has_sense(container-11,container.n.01)
has_sense(a-13,null)
has_sense(It-16,null)
has_sense(is-17,null)
has_sense(used-18,used.a.01)
has_sense(to-19,null)
has_sense(stir-20,null)
has_sense(serve-22,null)
has_sense(take-24,scoop.v.01)
has_sense(up-25,null)
has_sense(food-26,null)

// MANUALLY ANNOTATED:
has_sense(handle-14,handle.n.01)

is_a(container.n.01,container.n.01)
is_a(cutlery.n.02,cutlery.n.02)
is_a(handle.n.01,handle.n.01)
is_a(shallow.a.01,shallow.a.01)
is_a(bowl-shaped.s.01,bowl-shaped.s.01)
is_a(scoop.v.01,scoop.v.01)
is_a(null,null)

property(It-1,null)
property(is-2,null)
property(a-3,null)
property(piece-4,null)
property(cutlery-6,HYPERNYM)
property(a-8,null)
property(shallow-9,SIZE)
property(bowl-shaped-10,SHAPE)
property(container-11,null)
property(a-13,null)
property(handle-14,HASA)
property(It-16,null)
property(is-17,null)
property(used-18,null)
property(to-19,null)
property(stir-20,null)
property(serve-22,null)
property(take-24,null)
property(up-25,null)
property(food-26,null)
```

Listing 5.5: Syntactic Evidence Example - continued

Color name	H	S	V
pink.s.01	335	87	87
purple.s.01	290	87	87
blue.s.01	235	87	87
cyan.s.01	150	87	87
light-blue.s.01	175	87	87
green.s.01	115	87	87
green.n.01	115	87	87
yellow.s.01	50	87	87
yellowish.s.01	50	87	87
yellow.n.01	50	87	87
orange.s.01	20	87	87
orange.n.02	20	87	87
brown.s.01	20	87	97
red.s.01	0	87	87
blood-red.s.01	0	89	55
black.a.01	500	55	5
blackish.s.01	500	55	5
white.a.01	500	5	95
whitish.s.02	500	5	95
grey.s.01	500	5	50
greyish.s.01	500	5	50
gray.s.01	500	5	50
grayish.s.01	500	5	50

Table 5.1: Feature vectors for colors: each color symbol is assigned a vector containing the respective values for the corresponding color in the HSV color model.

Shape name	|edges|	|angles|	|faces|	subjective similarity
crescent.s.01	2	2	1	0
semicircular.s.01	2	2	1	1
curved.a.01	2	0	1	2
annular.s.01	2	0	1	4
ringlike.s.01	2	0	1	4
coil.n.02	2	0	1	5
coiling.s.01	2	0	1	5
rounded.a.01	0	0	1	6
roundish.s.01	0	0	1	7
egg-shaped.s.01	0	0	1	8
ellipse.n.01	0	0	1	8
flat.s.02	0	0	1	6
elliptic.s.01	0	0	1	8
pear-shaped.s.01	0	0	1	9
cylindrical.s.01	2	0	3	10
round.a.01	0	0	1	11
spherical.a.01	0	0	1	11.1
ball-shaped.s.01	0	0	1	11.2
circular.s.02	0	0	1	11
circular.n.01	0	0	1	11
octangular.a.01	8	8	1	14
hexangular.a.01	6	6	1	15
pentangular.a.01	5	5	1	16
quadrangular.a.01	4	4	1	17
square.n.01	4	4	1	17
square-shaped.s.01	4	4	1	17
rectangle.n.01	4	4	1	18
rectangular.s.01	4	4	1	18
orthogonal.s.03	4	4	1	20
boxlike.s.01	12	8	6	19
trapezoidal.a.01	4	4	1	21

Shape name	|edges|	|angles|	|faces|	subjective similarity
rhombic.a.01	4	4	1	22
triangle.n.01	3	3	1	24
triangular.s.01	3	3	1	25
pyramidal	5	5	5	26
wedge-shaped.a.02	9	6	5	27
cuneate.s.01	3	3	1	28)
conic.a.01	2	1	2	29
tapered.s.01	2	1	2	30)
asteroid.s.01	12	6	1	35
cordate.s.01	2	2	1	40)
convex.a.01	1	1	2	40
concave.a.01	3	3	2	40

Table 5.2: Feature vectors for shapes: each shape symbol is assigned a vector containing numeric values representing geometric characteristics.

Size name	n
dwarfish.s.01	0
bantam.s.01	1
little.s.03	2
shallow.a.01	2.5
small.a.01	3
modest.s.02	4
average.s.04	5
medium-sized.s.01	5
large.a.01	7
huge.s.01	8
grand.s.06	9

Table 5.3: Feature vectors for sizes: each size symbol is assigned numerical value according to its position in an ascending order from small to large.

CONFUSION MATRICES

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**4**	0	0	0	0	0	0	0.78	1.57	0	0	0	0	0
bowl.n.01	0	**2**	0	0	2.33	1	0.55	0	0	1.14	1	1.68	0	1.56
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0.73	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	0.43	0	0	0	0	0	0	0.86
knife.n.01	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**2**	1.5	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	0.75	**0**	0	0	0	0	0
pan.n.01	0	0.57	0	0	0.6	0	0	0	0	**2**	0	0.87	0	0.6
plate.n.04	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0
pot.n.01	0	0.84	0	0	0	0	0	0	0	0	0	**1**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

Figure 5.1: Confusion matrix for modeling II and configuration I of the formula template (with similarity).

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0
bowl.n.01	1	**2**	0	0	2	0	0	0	0	1	3	2	1	3
cherry.n.03	0	0	**0**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**0**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	0	0	0	0	**0**	0	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0
pot.n.01	0	0	0	0	0	0	0	0	1	0	0	**1**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0
spoon.n.01	3	2	4	4	2	4	4	4	3	3	1	1	3	**1**

(a) Confusion matrix for modeling I and configuration II of the formula template (no similarity, no closed world assumption).

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0
bowl.n.01	0.38	**2**	0	0	1.56	0	0	0	0	0.57	1.5	1.68	0.6	2.33
cherry.n.03	0	0	**0**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**0**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	0	0	0	0	**0**	0	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0
pot.n.01	0	0	0	0	0	0	0	0	0.36	0	0	**1**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0
spoon.n.01	1.2	1.56	1.6	1	1.6	3.43	2.29	1.52	1.14	1.8	0.76	0.78	1.89	**1**

(b) Confusion matrix for modeling I and configuration II of the formula template (with similarity, but no closed world assumption).

Figure 5.2: Confusion matrices for modeling I and configuration II.

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0
bowl.n.01	0	**0**	0	0	0	0	0	0	0	0	0	0	1	0
cherry.n.03	0	0	**0**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**0**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	1	0	0	0	**1**	1	0	1	1	0	0	0	0
knife.n.01	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**0**	0	1	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0
pot.n.01	4	3	4	4	4	3	3	4	3	3	4	**3**	3	3
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**

(a) Confusion matrix for modeling I and configuration III of the formula template (no similarity and no closed world assumption).

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0
bowl.n.01	0	**0**	0	0	0	0	0	0	0	0	0	0	0.6	0
cherry.n.03	0	0	**0**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**0**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	0.5	0	0	0	**1**	0.43	0	0.35	0.45	0	0	0	0
knife.n.01	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**0**	0	0.87	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0
pot.n.01	1.52	2.53	1.52	0.94	3.11	1.5	1.75	1.45	1.09	2.61	2	**3**	1.91	2.33
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**

(b) Confusion matrix for modeling I and configuration III of the formula template (with similarity, but no closed world assumption).

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**0**	0	0	0	0	0	0	1.57	0	0	0	0	0	0
bowl.n.01	0	**0**	0	0	0	0	0	0	0	0	0	1.68	0	0
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**4**	0	0	0	0	0.6	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	0.87	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	0.87	**2**	0	0	0	0	0	0	0.57
lemon.n.01	2.35	0	0	0	0	0	0	**0**	0	0	0	0	0	0
orange.n.01	0.78	0	0	0	0	0	0	1.5	**4**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**3**	0	0.87	0	0
plate.n.04	0	1	0	0	0	0	0	0	0	0	**2**	0.5	0	0
pot.n.01	0	1.68	0	0	0	0	0	0	0	0	1	**0**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**4**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**

(c) Confusion matrix for modeling I and configuration III of the formula template (with similarity and closed world assumption).

Figure 5.3: Confusion matrices for modeling I and configuration III.

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**0**	0	0.82	0	1.6	0	0.67	0	1.57	0.35	0	0.76	0.36	0.8
bowl.n.01	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
cherry.n.03	0	0	**0**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**0**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	0	0	0	0	**0**	0	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**0**	0	0	0
pot.n.01	0.38	1.68	0	0.47	0	0	0	0.73	0.36	0	0	**1**	1.27	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0
spoon.n.01	1.2	0.78	1.2	0.5	0	3.43	1.14	0.76	0.38	1.8	3.05	0.78	0.63	**2**

(a) Confusion matrix for modeling I and configuration I of the formula template (with similarity but no closed world assumption).

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**4**	0	0	0	0	0	0	0.78	1.57	0	0	0	0	0
bowl.n.01	0	**2**	0	0	0	0	0	0	0	0	0	0.84	0	0
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**4**	0	0	0	0	0.6	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	0.87	0	0	0	0	0	0	0
knife.n.01	0	0	0	0	0	0.87	**0**	0	0	0	0	0	0	1.14
lemon.n.01	0	0	0	0	0	0	0	**2**	1.5	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	0.75	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**3**	0	0.87	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**4**	0	0	0
pot.n.01	0	1.68	0	0	0	0	0	0	0	0	0	**2**	0	0
spatula.n.02	0	0	0	0	0	0	1.39	0	0	0	0	0	**4**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**

(b) Confusion matrix for modeling I and configuration I of the formula template (with similarity and closed world assumption).

Figure 5.4: Confusion matrices for modeling I and configuration I.

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**4**	0	0	0	0	0	0	1	2	0	0	0	0	0
bowl.n.01	0	**2**	0	0	3	2	2	0	0	2	2	4	1	2
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	0	0	0	0	0	0	0	2
knife.n.01	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**2**	2	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	1	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	1	0	0	0	0	**2**	0	0	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0
pot.n.01	0	2	0	0	0	0	0	0	0	0	0	**0**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

(a) Confusion matrix for modeling II and configuration II of the formula template (no similarity).

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**4**	0	0	0	0	0	0	0.78	1.57	0	0	0	0	0
bowl.n.01	0	**2**	0	0	2.33	1	1.09	0	0	1.14	1	3.37	0.6	1.56
cherry.n.03	0	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	0	0	0	0	0	0	0	1.71
knife.n.01	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**2**	1.5	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	0.75	**0**	0	0	0	0	0
pan.n.01	0	0	0	0	0.6	0	0	0	0	**2**	0	0	0	0
plate.n.04	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0
pot.n.01	0	1.68	0	0	0	0	0	0	0	0	0	**0**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**

(b) Confusion matrix for modeling II and configuration II of the formula template (with similarity).

Figure 5.5: Confusion matrices for modeling II and configuration II.

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**3**	0	0	0	0	0	0	2	0	0	0	0	0	0
bowl.n.01	0	**2**	0	0	3	1	2	0	3	4	3	4	2	2
cherry.n.03	1	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	0	0	0	0	0	0	0	1
knife.n.01	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	2	**1**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0
plate.n.04	0	1	0	0	0	1	0	0	0	0	**1**	0	0	0
pot.n.01	0	1	0	0	1	0	0	0	0	0	0	**0**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**

(a) Confusion matrix for modeling II and configuration III of the formula template (no similarity).

Prediction/Ground Truth	banana.n.02	bowl.n.01	cherry.n.03	coffee.n.01	cup.n.01	fork.n.01	knife.n.01	lemon.n.01	orange.n.01	pan.n.01	plate.n.04	pot.n.01	spatula.n.02	spoon.n.01
banana.n.02	**3**	0	0	0	0	0	0	1.57	0	0	0	0	0	0
bowl.n.01	0	**2**	0	0	2.21	0.51	1.12	0	1.12	2.34	1.54	3.2	1.23	1.47
cherry.n.03	0.82	0	**4**	0	0	0	0	0	0	0	0	0	0	0
coffee.n.01	0	0	0	**4**	0	0	0	0	0	0	0	0	0	0
cup.n.01	0	0	0	0	**0**	0	0	0	0	0	0	0	0	0
fork.n.01	0	0	0	0	0	**2**	0	0	0	0	0	0	0	0.9
knife.n.01	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0
lemon.n.01	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0
orange.n.01	0	0	0	0	0	0	0	1.5	**1**	0	0	0	0	0
pan.n.01	0	0	0	0	0	0	0	0	0	**0**	0	0	0	0
plate.n.04	0	0.51	0	0	0	0.8	0	0	0	0	**1**	0	0	0
pot.n.01	0	0.8	0	0	0.67	0	0	0	0	0	0	**0**	0	0
spatula.n.02	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0
spoon.n.01	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**

(b) Confusion matrix for modeling II and configuration III of the formula template (with similarity).

Figure 5.6: Confusion matrices for modeling II and configuration III.