STS
Software
Technology
Systems

TUHH
Hamburg University of Technology

Grounding Words to Objects:

A Joint Model for Co-reference and Entity Resolution Using Markov Logic for Robot Instruction Processing

Diplomarbeit

Florian Meyer

Prüfer der Diplomarbeit:	1. Prof. Dr. rer. nat. habil. Ralf Möller
	(TU Hamburg Harburg)
	2. Univ.-Prof. Michael Beetz, PhD
	(TU München)

Eidesstattliche Erklärung

Ich, Florian Meyer, geb. am 25.10.1985, versichere hiermit, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen oder anderen Quellen entnommen sind, sind als solche eindeutig kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form noch nicht veröffentlicht und noch keiner Prüfungsbehörde vorgelegt worden.

Hamburg, ...

Acknowledgements

I would like to thank Daniel Nyga for his support during the creation of this work. Moreover, I would like to thank everyone of the former IAS group at the TU-München for their help and for making my stay in their group worthwhile. Letting me learn about many aspect of the robotic research was rewarding far beyond the scope of this.

Additionally, I would like to thank my two supervisors Prof. Beetz and Prof. Möller. Prof. Beetz for accepting me as a student and integrating me flawlessly into his group and Prof. Möller for supporting the thesis by being the first supervisor.

Abstract

This work is concerned with the development of a novel approach to language grounding in the context of autonomous robotics. A probabilistic first order knowledge base is used to build a database for action specific background knowledge for everyday manipulation tasks. It is shown that such a knowledge base can be used to find items in the robot's current belief state of the world which are necessary for a successful task execution. This can be achieved for objects that are specifically named in the text and objects that are expected by an action specific model but that are not explicitly named. The developed models are tested in several experiments and are critically analysed.

Contents

List of Resources

0ptAlgorithms

Chapter 1

Introduction

0ptMotivation

One of the major challenges in the field of autonomous robotics is the intuitive interaction between the operator and the robot. One approach to this problem is to use natural language and to interact with the robot with regular, everyday instructions. A scenario for such an interaction could be in a kitchen setting where the robot has to do tasks that are everyday activities and can be successfully accomplished by humans without great difficulty, like cooking dishes or setting the table. In order to have a natural feeling while interacting with the robot, the human operator should be able to give general, even abstract task instructions like “Set the table” or “Make me some pancakes”.

For the autonomous robot being able to process such general task instructions it has to have background knowledge about the instructions. Fortunately, for many of the everyday tasks that are encountered in a kitchen setting there are detailed task instructions available on the internet[1]. These databases include many recipes and descriptions of many other everyday tasks. This huge amount of data has proven to be exploitable in the robotics context [**?**]. Nonetheless, the downside of those tasks descriptions is the fact that they are intended for other humans. Henceforth, there is a need to harness these task descriptions by transforming the natural language into a representation that can directly be used further by robotic execution engines.

Instructions that are intended for humans are inherently underspecified [**?**]. For example, the sentence *“Put pasta into a pot.”* is lacking a lot of crucial information the

[1]e.g. www.ehow.com, www.wikihow.com

robot needs in order to successfully accomplish the task. First, the robot needs to disambiguate what the semantic meaning of all the different words is. For example, the semantic sense of the word “*Mix*”: It could either refer to the verb “*to mix*” that would be referenced in the context of an instruction like “*Mix pasta and water.*” or as the noun “*the mix*” that could occur in a sentence like “*Keep the mix refrigerated.*”. Moreover, not all actions necessary for a robot to execute are explicitly mentioned in a set of instructions that is intended to be processed by humans. For example, to open the box of pasta and take out the contents before adding the pasta to a pot. Those implicit instructions also depend on the state of the world that the robot operates in and consequently are hard to put in a general web instruction that is intended to be general enough to be understood by everyone and be applicable in every kitchen.

It has been argued that everyday tasks require a huge amount of background knowledge in order to be executed successfully [**?**]. This background knowledge base needs to include data on many different levels. On a high level of abstraction this knowledge base needs to contain data about the preferences of the operator[2] or the habits[3] [**?**]. On a medium level, this knowledge base could include knowledge about the objects that are usually used for specific tasks[4] and on a lower level this could also include knowledge about the relationships between the different objects in the context of a specific task, for example, the geometric relationships between objects in a kitchen setting in the context of a particular task. Consider the following instruction: ‘*Put the cup on the table.*’. The instruction expects to use a cup that is present in the kitchen and not yet on the table. Consequently, there are certain patterns that are assumed to hold with those instructions and this could also be saved in such a background knowledge base.

One of the fundamental problems in the context of underspecified instructions is the identification of the objects that are expected to be used in the context of an instruction. These can be stated explicitly in the text or implicitly be expected to be used for certain tasks. This knowledge would be expected to be available in some background knowledge. The objects that are present in a set of task instructions can be repeatedly referred to in a text with different names or implicitly within different actions. Being able to identify, which object is referred to, is therefore a key challenge in processing

[2]e.g. where does the operator want to sit on the table?
[3]e.g. how many pancakes does the operator eat?
[4]e.g. use the middle sized pan to make pancakes

robotic task instructions and make them useful for the autonomous robot. The task of determining if two words refer to the same real world objects is generally known as coreference resolution. To find the real world entity that belongs to a word is known as entity resolution. In the context of task execution the underspecification plays a central role while developing solutions that can process the instructions. As will be shown in the next section, current available solutions for coreference and entity resolution do not provide mechanisms to handle underspecified instructions.

0ptProblem Description

Instructions for everyday activities in natural language are highly underspecified [**?**]. Everyday instructions usually mention objects implicitly because the knowledge of the human provides the necessary background information about the objects involved. For example in the short set of instructions:

Put pasta into a pot.
Add water.
Cook for 10 minutes.
Serve on a plate.

In the second instruction the verb '*Add*' requires some knowledge about the place where to add water. This location is the pot from the first sentence. Consequently, a model is needed that naturally models this kind of underspecified instructions.

The Probabilistic Robot Action Core (PRAC) [**?**] formalism tries to resolve this kind of underspecification by introducing roles. The PRAC formalism is formally introduced in Section 2.1 but the main ideas will be presented here to give the reader the opportunity to understand the context of this work. For the most important action verbs there are specific models that have roles which parametrize the action. A role is defined to fulfil some function in this action. For the above example the resulting representation is provided in Figure 1.1. Each word is assigned its semantic role in the context of the action it appears in using probabilistic inference. This is indicated by the arrows pointing to the green nodes. The sentences can be treated independently as it is assumed that the information for one action can be found in the sentence that the verb appears

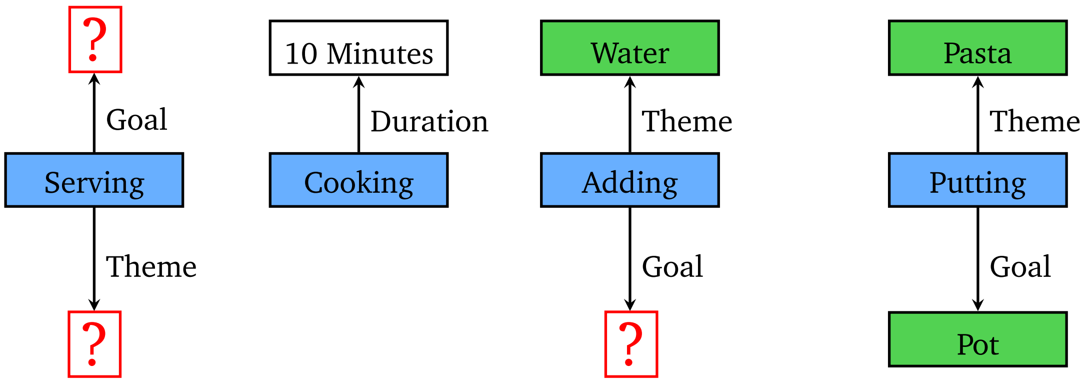


Figure 1.1.: *The PRAC roles for the actions from the example.*

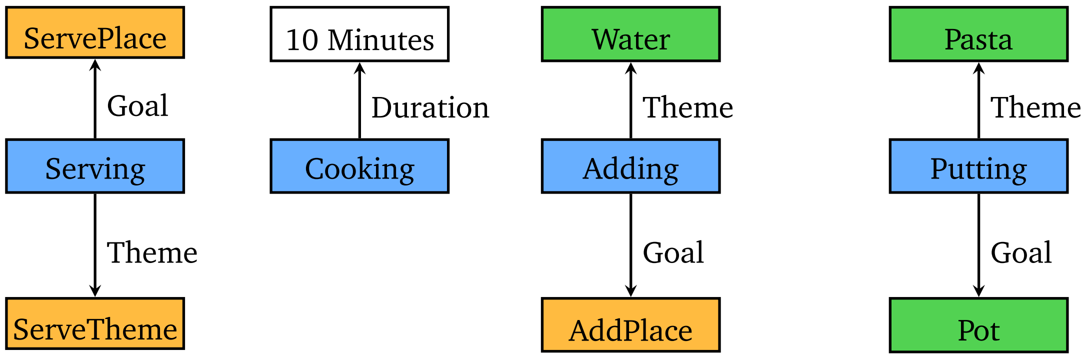


Figure 1.2.: *Three virtual words (orange) are introduced to enrich the model.*

in. As can be seen, the presented model in Figure 1.1 is only a reduced version since in a real application the robot needs a lot more information. However, the presented information is enough to understand the principles explained in this work.

Figure 1.1 additionally exemplifies that for three roles, no word in the instruction can be found. If a role cannot be filled with a word in the sentence then a *virtual word* is introduced, i.e. a word that belongs to the sentence but is not explicitly stated. The role is then assigned to this newly introduced word. Figure 1.2 shows how three virtual words are introduced and the roles are assigned to those words.

Within the context of this semantically enriched version of the instruction set, this work aims to address the problem of coreference and entity resolution. Coreference aims to find all virtual and actual words that represent the same real world entity.

Entity resolution tries to find all real world representations of the coreference clusters. This is shown in Figure 1.3. A '*Coreference*' edge indicates that two words (including virtual words) are in a coreference relationship. The entity resolution is indicated by the '*Grounded*' relationship. The images present objects that exist in the robot's current belief state of the world. Moreover, the objects in the world can be in different relationships to each other. In the example the '*Pasta*' is in a geometric '*on*' relationship with the '*kitchen table*'.

Coreference and entity resolution in the robotics context are two closely connected tasks. Both deal with the identification of the objects in a text. Usually, those two tasks are treated separately as non relational learning methods do not allow the joint inference over both tasks naturally. Nonetheless, these two task are related in a way that the information about the existence of one is evidence for the other and vice versa. One such example for the set of instructions mentioned above could be: The word '*pot*' in the first sentence is implicitly referred to in the second instruction because it represents the location the water is added to. If this coreference can be inferred then it can also be inferred that they represent the same real world object. Hence it is beneficial to treat those two tasks together in one model.

In conclusion, the task is to find all coreference relationships between words, whether they are virtual or explicitly mentioned words. Moreover, to find the most likely match of an object mentioned in a text with an object in the robot's current belief state of the world.

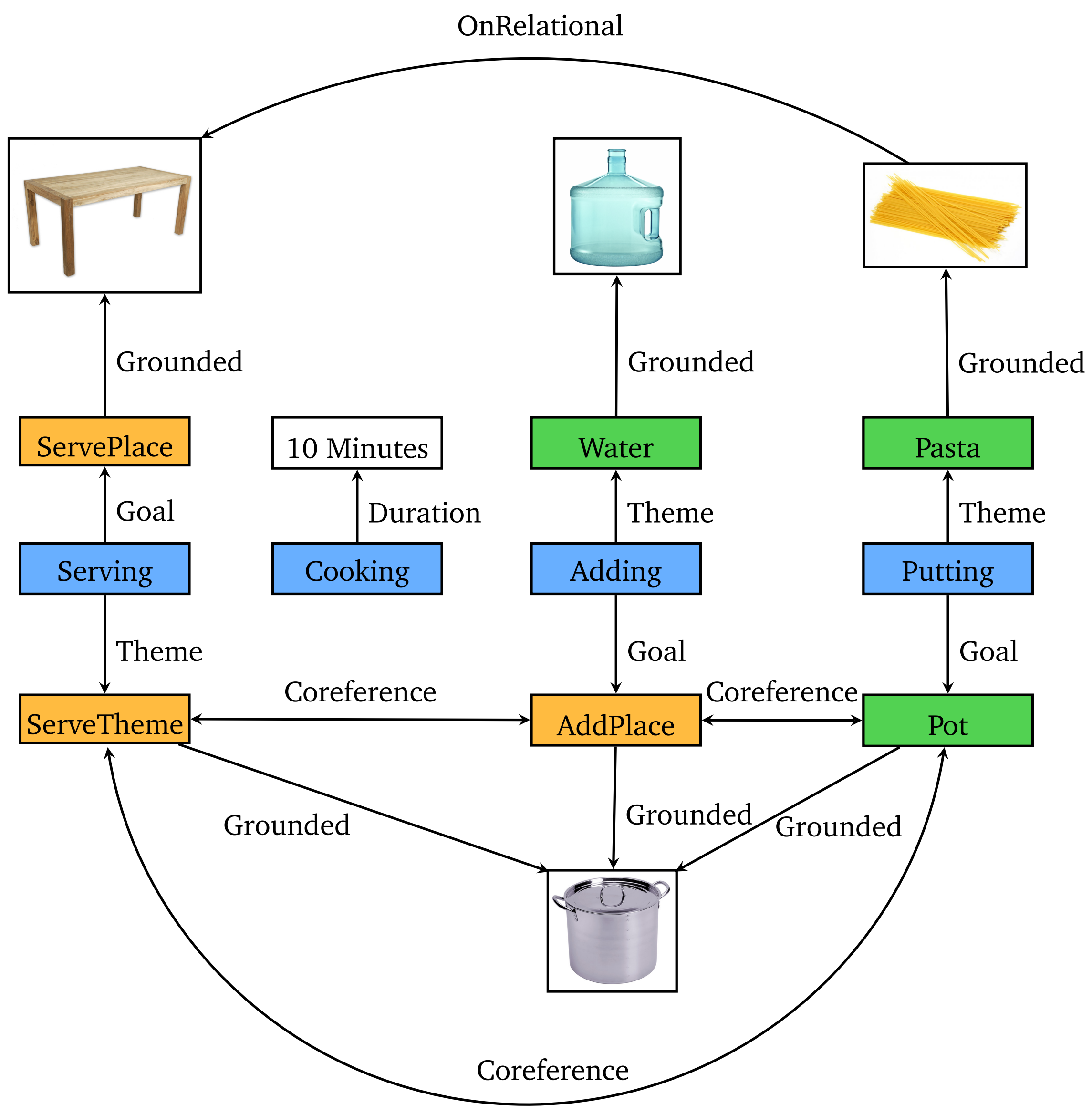


Figure 1.3.: *The relationships of the different entities encountered in a short set of instructions.*

0ptRelated Work

Coreference and entity resolution are both terms that were coined in the context of classical Natural Language Processing (NLP) tasks. Coreference in this context is defined as finding the words in a text that refer to the same object. For example, the sentences: "*John puts a glass on the table. Now he drinks from it.*" In this short example a coreference system is to find that the word "*he*" in the second sentence refers to the word "*John*" in the first sentence and the "*it*" in the second one to "*glass*" in the first sentence. A good overview on the history of coreference approaches is provided in [**?**]. A recent state of the art implementation is described in [**?**] where many syntactic and semantic features are extracted and processed in a sieve like approach, where the strongest indicators are applied first and less strong indicators later. This approach includes the use of semantic distances in the WordNet taxonomy, semantic information from the Wikipedia infoboxes[5] and Freebase records[6]. Clusters of coreferent words are created and mentions can either be added to a cluster or be left out. In this fashion a set of 13 sieves are applied sequentially. However, this system has been trained with annotated newspaper articles and as a result usually expects grammatically correct sentences. Moreover, certain sentence structures are seldom encountered in such newspaper articles, e.g. imperatives that are frequently encountered in robot instructions. In [**?**] a joint coreference resolution approach using Markov Logic (ML) is introduced. This approach differs from traditional approaches as it does not look at the coreference for pairs of mentions but classifies all mentions jointly. Additionally, this system uses an unsupervised approach and consequently does not rely on annotated training data. It shows that this unsupervised approach outperforms existing unsupervised coreference systems and is comparable to supervised ones that do not make use of joint models. The work is based on the work described in [**?**]. The work described is also unsupervised but lacks a joint approach. Evidence for this model is also based on syntactic features that can be extracted from the text.

Entity resolution on the other hand is the task to find entities that exist in the real world in intra text environments. For example, the fact that "*BMW*" always refers to the company "*BMW*" in multiple independent documents. In this area as well a lot of work has been done. One work [**?**] uses ML to match entities that reside in different

[5]www.wikipedia.com
[6]www.freebase.com

databases. In particular, the work tries to match scientific citations. The model is an extension to the model proposed by Fellegi and Sunter [**?**] and uses ML to do joint entity resolution by eliminating the independent and identically distributed (i.i.d.) assumption. This is one example on how to approach the classical NLP entity resolution problem.

Other works focus only on particularly hard to detect features like *Noun Genders* [**? ?**] or to identify the pleonastic "*it*" [**?**]. Those works can be beneficial to entity as well as coreference resolution. It is important to point out, though that determining the gender of nouns is in general not possible in the English language and is only applicable to entities like certain humans or animals (in case of animicity). Other languages like French or German, however, would greatly benefit from those works in coreference and entity resolution settings. As this describes a problem that fundamentally differs from the one dealt with in this work a further evaluation is out of the scope of this work.

In a robotics setting the robot needs to get information about the meaning of the words and in particular what those words imply for the robot in its current context, i.e. the world it resides in. Processing natural language and making its content accessible for robots is also called *Semantic Language Processing (SLP)* or *Language Grounding (LG)*. The former parses language in order to extract the meaning of the text or sentence and its components while the latter goes further and grounds the language into the capabilities of the robot and its environment. For example, a robot needs to have knowledge of the intended meaning of a verb in a sentence. But in addition to having information about what the verb means, it needs to have a mapping from the intended meaning of the word to the correct actions in the context that the robot is in at the moment. The explanations given here are to be used as rough outlines of the fields as the terms introduced are not strictly defined.

This work is at the intersection of SLP and LG, meaning that the sense of the words has already been allocated and the goal is finding the object that are referred to in the sentences, in the perceived world of the robot. In this approach there is no translation of the sentence into a robot plan or any other plan language but into the PRAC [**?**] formalism. This representation can be used to build high level plans that can be used in an execution engine like the one described in [**?**].

Prior work in LG mostly deals with grounding directions that are given by a human to a robot. The approach described in [**?**] accomplishes this by using techniques from statistical machine translation where parts of the sentence are paired with manually annotated λ-expressions. In this way entire sets of directions can be translated into a nested λ-expression. This approach is very limited in its applicability and requires a lot of manually annotated training data. Moreover, its generalization behaviour over unknown, i.e. not previously seen objects in the training data, is questionable.

Another approach [**?**] also deals with directions given by a human to a robot. The Generalized Grounding Graphs (G^3) framework is developed, in which natural language instructions are converted into factor graphs, where each factor is connected to three nodes. One of the nodes corresponds to an expression in the text and one to an object in the world. The third node is *True* if the two other nodes match. This factor graph representation corresponds to a visualisation of the Spatial Description Clause (SDC) [**?**]. This extends the schema just presented to spatial relations and hence to more than just objects, e.g. "verbs" and "places" can be represented. Since the factor graph can be translated into a Markov Network it is evident that this propositional approach is rather limited in its expressiveness. For example, the types (e.g. "Place", "Objects", etc.) need to be known in advance and the extension is difficult. However, the way the different parts of the factor graph can be connected shows some resemblance with the PRAC approach.

Further, a different work [**?**] uses a combination of λ-calculus and Combinatorial Categorical Grammars where combinatorial rules determine how the translation process is taking place. For each instruction a *goal* and an *action* is expected. Although the system can run in real-time and can create usable robot plans, it is not able to handle uncertainty in the interpretation of word senses. It does not generalize over the words and requires that the semantics of each word are annotated in the database used.

In case of entity resolution the classical NLP understanding of this problem is not applicable to grounding the entities of the text into the perceived world of the robot. Furthermore, the entity resolution problem from the NLP context does not take the properties of those entities into consideration that are not mentioned explicitly in texts, like the location of an object. Additionally, only those objects are searched for that are explicitly stated in the instruction but when dealing with incomplete information, this is not sufficient. Coreference is mainly dealt with in the classical NLP context where mostly syntactic information is used. However, in the robotics context where a lot of semantic information is available and needed, those approaches need to be further refined in order to find objects in different instructions that are not explicitly mentioned. In general, all of the presented approaches do not address the problem of underspecified instructions.

In conclusion, this short summary makes clear that there is a need for new methods that can solve the problem of underspecification in everyday task descriptions since today's methods do not address this problem. This work addresses the problem of finding a new approach to coreference and entity resolution that naturally handles underspecified instructions in a robotics context.

0ptContributions

This work presents a novel approach to language grounding using a probabilistic first order knowledge base. The work addresses the problem that underspecified instructions pose by using virtual words and roles. It is shown how Markov Logic can be used as a means to provide such a knowledge base and how the formalism can be exploited to infer missing information. A system for coreference and entity resolution is developed. Moreover, several experiments will show the applicability of the approach presented here. This includes to find the words that are in a coreference relationship within a text and identify the objects that can be used in the current belief state of the robot.

0ptOutline

This work is divided into four chapters. It begins with introducing the necessary prerequisites that are elementary to the understanding of this work. First, First Order Logic and Markov Networks are covered in Section 2.2 and Section 2.3. Those two methods are needed for the introduction to statistical relational learning in Section 2.4 and concluding with Markov Logic in Section 2.4.2 which is the method mainly used in this work for the implementation. Moreover, a short introduction to Probabilistic Robot Action Cores is given in Section 2.1 to explain the context in which this work has been developed. Chapter 3 provides details on how the different models for co-reference and entity resolution are designed. Section 3.2 continues with the implementation in Markov Logic. Experiments are conducted and results are being discussed in Section 3.3 and Section 3.4. The model used in analysed in detail in Section 3.5. The work ends with an outlook on potential improvements and a brief summary of the results obtained in Chapter 4.

Chapter 2

Prerequisites

0ptProbabilistic Robot Action Cores

This work has been developed in the context of the ongoing effort at the Intelligent Autonomous Systems (IAS) group at the Technische Universität München (TUM) to provide knowledge bases for autonomous robots. The PRAC formalism exploits freely available on-line resources and methods from Statistical Relational Learning (SRL) to build action specific background knowledge that is available to the robot in order to be able to execute everyday tasks that are easily handled by humans. The concept of the PRAC is derived from the concept of the semantic core [**?**] that was introduced in psychology to explain how the human mind is able to understand and execute highly underspecified everyday instructions.

A PRAC is an action specific probabilistic knowledge base that models an abstract event type by assigning an action role to each entity that is affected by the action verb. An action verb in this case is a word in the instruction that results in an action by the robot. Therefore the knowledge base provides knowledge about all the information that the robot needs on a high level to successfully execute the task. This knowledge ranges from identifying the objects that are needed in the instruction to geometric constraints between these objects. Moreover, a PRAC provides knowledge about preferences of users and can save knowledge about the environment the robot acts in. For example the action "to place something somewhere": In the context of this instruction it is important to know where something needs to be placed. And of course what needs to be placed. This is already very context dependent. In the case that a hot pot is to be placed on a wooden table, the operator would expect the robot to know that the pot is supposed to be placed on top of same protective layer between the hot pot and the table. Many of those situations arise where a lot of context dependent background knowledge is needed, especially since most natural language instructions are highly underspecified. In other words, a PRAC is a template for an action that is parametrized by the context under which the action is to be executed.

Formally a PRAC is defined in definition 2.1.1.

Definition 2.1.1 ([?]) *A Probabilistic Robot Action Core is a conditional probability distribution* $P(\mathbf{R} \times \mathbf{T} \times \mathbf{S} | \sqsubseteq, \preceq)$, *where*

$\mathbf{R}$	*is the set of all action-specific relations*
$\mathbf{T}$	*is the set of all action verbs*
$\mathbf{S}$	*is the set of all class concepts*
$\sqsubseteq$	*is a taxonomy relation over* $\mathbf{S}$
$\preceq$	*is a mereological relation over* $\mathbf{S}$

The set of *class concepts* is a set that contains concepts of objects. $\sqsubseteq$ defines relationships between all the concepts and thus induces a taxonomy. As will be shown this is very important for the generalization properties of the PRAC formalism over new unseen objects, i.e. objects not seen in the training phase. Mereological relationships can further help make the correct predictions.

Since the PRAC is defined as a Conditional Probability Distribution (CPD) any probabilistic inference is possible given the context of the current instruction and the currently available knowledge.

To fill the actions specific knowledge base multiple sources of knowledge can be used, for example:

- Annotated textual data
- Annotated video data
- Simulator data

Each of these sources provides different information about the context of actions. Video and simulator data could provide geometric relations between objects that are usually not available in textual data.

In order to process a natural language instruction and transform it into the PRAC representation, many different steps are necessary. The details of the implementation are left for Section 3.2.

0ptFirst Order Logic

First Order Logic (FOL) is a powerful method to model knowledge in relational domains. It generalizes propositional logic to a higher level of abstraction where reasoning over sets of objects is possible. FOL extends propositional logic with the use of quantifiers that allow to make a statement over a domain of symbols.

The following short introduction to the basics of FOL is closely inspired by the introduction given in [**?**].

A set of sentences or formulas in first-order logic is considered a first-order knowledge base. Formulas are built using four different symbols: *constants*, *variables*, *functions* and *predicates*. *Constants* are symbols that represent the entities that exist in the domain of discourse and may be typed. This could be words, furniture pieces or humans. *Variables* are place holders for objects of an entire domain and they may be typed, i.e. assigned to one particular domain. A *Function* symbol is a representation of an object that is in a relationship to another object without having to name the former, e.g. '*FatherOf(Bob)*'. *Predicates* represent relations of objects in the domain or assign attributes to objects. An *interpretation* or *possible world* is used to specify which symbols are represented by which *functions*, *objects* and *relations*.

Formulas are built using four logical connectives as well as quantifiers and recursively connecting *atomic formulas*. An *atomic formula* is is a predicate symbol applied to a tuple of *terms*. A *term* is a logical expression that refers to an object. Thus, constants are terms but functions are terms as well. The logical connectives are:

- '$\neg$' - negation
- '$\wedge$' - conjunction
- '$\vee$' - disjunction
- '$\Rightarrow$' - implication

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
True	*True*	*False*	*True*	*True*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*	*False*
False	*True*	*True*	*False*	*True*	*True*	*False*
False	*False*	*True*	*False*	*False*	*True*	*True*

Table 2.1.: *The truth table for the FOL connectives.*

- '$\Leftrightarrow$' - equivalence

The truth values for the connectives can be taken from Table 2.1.

For quantification, two operators are available

- '$\forall$' - universal quantification
- '$\exists$' - existential quantification

A formula $\forall x F_1(x)$ is '*True*' if F_1 is '*True*' for all objects x in the domain of x. $\exists x F_1(x)$ is '*True*' iff there exists at least one object in the domain of x where F_1 is '*True*'. A complete grammar of the syntax in the Backus-Naur form is provided in [**?**]. A *ground term* is a *term* that has no variables. A *ground atom* is an atomic formula where each argument is a *ground term*. A *possible world* assigns a truth value to each possible *ground atom*. A knowledge base intuitively is implicitly conjoint just like a regular formula. A formula F_1 is called *satisfiable* iff there exists at least one world in which F_1 is '*True*'. A formula F_1 is said to entail another formula F_2 iff F_2 is '*True*' in all worlds where F_1 is '*True*' and this is denoted as $F_1 \models F_2$.

As can be seen from this short introduction, FOL is a formalism where relationships between classes and object instances can easily be represented. However, the formulas that are defined in the knowledge base are expected to always be '*True*'. In real applications this is very hard to guarantee, even if noise in the data could be ignored. As a consequence, models that naturally model uncertainty are desirable.

In the next section, Markov Networks are introduced that belong to the class of probabilistic graphical models and provide an intuitive way of modelling uncertainty.

0ptMarkov Networks

The Markov Network (MN) formalism is a widely used formalism to model uncertainty. MNs or Markov Random Fields are undirected graphical models of the joint distribution of a set of random variables. In its graphical form it encodes independence assumptions and is consequently a tool for engineers to model complex domains easily.

For variable sets $\mathbf{A},\mathbf{B}$ and $\mathbf{C}$, the expression $\mathbf{A} \perp \mathbf{B} \mid \mathbf{C}$ states that the variables $\mathbf{A}$ are independent of the set $\mathbf{B}$ if conditioned on set $\mathbf{C}$. With this notation, MNs are defined in definition 2.3.1.

Definition 2.3.1 ([?]) *A Vector $\mathbf{X}_V$ with random variables as elements, indexed by a vertex set $\mathbf{V}$, is a Markov Network over an undirected graph $G = (\mathbf{V}, \mathbf{E})$, with a set of edges $\mathbf{E}$, if and only if each random variable X_v, conditioned on its neighbours, is independent of all other elements of $\mathbf{X}_v$ with $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ and $v_1, v_2 \in \mathbf{V}$, $(v_1, v_2) \in \mathbf{E} \Longleftrightarrow \forall (v_2, v_1) \in \mathbf{E}$:*

$$(\forall v \in \mathbf{V}) : X_v \perp \{X_u : u \neq v, (u,v) \notin E\} | \{X_u : (v,u) \in E\} \tag{2.1}$$

However, definition 2.3.1 lacks an explanation of how a probability distribution is defined over a MN. For this, the result of a famous theorem in MNs is needed.

Theorem 2.3.1 (Hammersly-Clifford[?]) *A probability distribution P over $\mathbf{X} = \{X_i, i \in \mathbf{V}\}$ satisfies the independence assumption made in definition 2.3.1 for a graph $G = (\mathbf{V}, \mathbf{E})$ if and only if $P(\mathbf{X})$ factorizes according to the set of cliques $C \in \mathbf{C}$ in G, i.e. can be represented as a proportional product of the cliques in G*

$$P(\mathbf{X}) \propto \prod_{C \in \mathbf{C}} \phi_C \tag{2.2}$$

where ϕ_C is a non negative, real valued function that depends only on the variables $\mathbf{X}_C = \{X_{v_1}, \ldots, X_{v_n}\}$ and a clique is a fully connected sub graph of G.

The potentials ϕ_C assign each state of the clique a real, non negative value. In order to get legal probabilities, the product of the clique potentials needs to be normalized by the partition function given in Equation (2.3).

$$Z = \sum_{\mathbf{X} \in \mathbf{X}_V} \prod_{C \in \mathbf{C}} \phi_C \tag{2.3}$$

The computation of the partition function is very hard, as can be seen from the formula. Fortunately it is not always the case that is actually necessary to compute it exactly. Approximation is often sufficient or in the case of Most Probable Explanation (MPE) inference no computation is needed, as pointed out in Section 2.4.3.

Defining the probability distribution directly with Equation (2.2) requires to model each state of the entire distribution explicitly. This ends up in a table of a size that is exponential in the number of variables in a clique. However, in many cases the MN structure can be exploited to simplify the modelling of a MN. The clique potentials can be transformed into one or many feature functions $f(X)_C$ with weights w_C.

$$\begin{aligned} & P(\mathbf{X}) && = \frac{1}{Z} \prod_{C \in \mathbf{C}} \phi_C \\ \Leftrightarrow \quad & P(\mathbf{X}) && = \frac{1}{Z} \prod_{C \in \mathbf{C}} \exp(\ln \phi_C) \\ \Leftrightarrow \quad & P(\mathbf{X}) && = \frac{1}{Z} \exp\left(\sum_{C \in \mathbf{C}} \ln(\phi_C) \right) \end{aligned} \tag{2.4}$$

A feature f_c is defined as

$$ln(\phi_C(\mathbf{X}_C)) = w_C \cdot f_C(\mathbf{X}_C)$$

and hence the *log-linear* model can be formulated as

$$P(\mathbf{X}) = \frac{1}{Z} \exp\left(\sum_{C \in \mathbf{C}} w_C \cdot f_C(\mathbf{X}_C) \right) \tag{2.5}$$

In the simplest case there is one feature function for each state of each clique, but in the best case, only one feature for an entire clique is needed. Thus modelling a MN with the log-linear model can save time for the designer as well as save memory since the value for each state of the clique can be computed on demand and does not need to be saved explicitly. It is important to note that the feature function can be any arbitrary function that is defined to create a mapping over a set of variables into $\mathbb{R}$.

In the next section, this MN formalism is used to combine probabilistic models as well as FOL models with the use of Markov Logic Networks (MLNs).

0ptMarkov Logic

0ptIntroduction

Represent, reason and learn in environments that are characterized by data that has a complex relational structure is a very active field of research [**?**]. The research is motivated by the fact that most data that is encountered in many applications like NLP has many interdependencies. Methods are needed where this relational structure can be modelled and where uncertainty can be incorporated naturally.

As an example the semantics of the words can be used. The semantics of the different words in texts have strong dependencies to the semantics of other words. For example, in the sentence “Mix A with B.”. The word “Mix” can have one of multiple senses, one as the noun “the mix” or the verb “to mix”. The other words of the sentence specify how the word needs to be interpreted to give meaning in that context. Motivated by the essentially ubiquitous existence of relational data, methods are needed where relationships can be modelled in a compact and mathematically sound manner. However, since not all evidence that is needed to make such decisions can be acquired without any noise, uncertainty needs to be an integral part of the formalism.

FOL provides a very intuitive way to model absolute statements about the relationships between classes of objects. However, FOL only allows the definition of hard formulas and is limited in that matter. In the case of uncertainty FOL is not applicable because the statements made in FOL must not contain any contradictions.
Uncertainty on the other hand is handled very well by probabilistic graphical models like Bayesian Networks [**?**] or Markov Networks, as introduced in Section 2.3 [**?**]. Those models are very popular within the research community due to their easily to understand semantics as well as the possibility to graphically model the systems.

As a consequence a combination of FOL and graphical models is desirable bringing together the easy to model relationships and to include uncertainty. One such approach is ML which is further introduced in this chapter. ML allows to model a system in a first order language that compiles those first order formulas into MNs. ML is therefore a template language for MNs. In the next sections ML is introduced as well as some inference and learning techniques that can be applied with the resulting MNs.

0ptFormal Definition

ML has been developed by Pedro Domingos[1] at the university of Washington. A MLN is defined as follows:

Definition 2.4.1 *[?][p.12]*

A Markov Logic Network (MLN) $\mathbf{L}$ *is a set of pairs* (F_i, w_i) *where* F_i *is a formula in First Order Logic (FOL) and* w_i *is a real number. Together with a finite set of constants* $\mathbf{C} = \{c_1, \ldots, c_{|\mathbf{C}|}\}$*, it defines a Markov Network (MN)* $M_{\mathbf{L},\mathbf{C}}$ *as follows:*

- $M_{\mathbf{L},\mathbf{C}}$ *contains one binary node for each possible grounding of each predicate appearing in* $\mathbf{L}$*. The value of the node is* 1 *if the ground predicate is* $true$*,* 0 *otherwise.*
- $M_{\mathbf{L},\mathbf{C}}$ *contains one feature for each possible grounding of each formula* F_i *in* $\mathbf{L}$*. The value of this feature is* 1 *if the ground formula is* $true$*, and* 0 *otherwise. The weight of the features is the* w_i *associated with* F_i *in* $\mathbf{L}$*.*

As can be seen from Definition 2.4.1 there is an edge between two nodes in the grounded MN if and only if the corresponding predicates appear together in at least one formula. Moreover, the definition also implies the joint distribution over the set of possible worlds that is defined as

$$P(\mathbf{X} = x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{|\mathbf{L}|} w_i n_i(x)\right) = \frac{1}{Z} \exp\left(\sum_{j=1}^{|\mathbf{G}|} \widehat{w}_j \widehat{f}_j(x)\right), \tag{2.6}$$

[1]http://homes.cs.washington.edu/ pedrod/

where **X** is the set of possible worlds. Equation 2.6 states that the probability of a world depends on the number of true groundings of all formulas and their weights. $n_i(x)$ is the number of true groundings of the i-th formula. $\widehat{f_j}$ is the j-th grounded formula from the set **G** that contains all grounded formulas and $\widehat{w}_j$ the corresponding weight. A feature is defined for each grounded formula that has the value *1* if the grounded formula is *True*.

An algorithm to get the groundings is provided in [**?**][p.13].

0ptInference

Working with ML gives rise to the possibility to make use of the entire range of algorithms that have been developed to do inference in MNs. Basically those algorithms can be categorized into two groups:

- exact algorithms
- approximate algorithms

Due to the fact that all exact inference in MNs is always #P-hard and hence intractable [**?**], usually approximate algorithms are employed. In case that only the most probable variable configuration is needed, MPE inference can be used. This is also an NP hard problem but with good heuristics it can often be solved exactly in reasonable time.

The next section presents how an MLN can be transformed into a Weighted Constraint Satisfaction Problem (WCSP) and then how fast MPE inference on the resulting WCSP can be applied. This is often sufficient in a robotics setting since the robot needs absolute statements about the objects to pick up or tools to use. Additionally, one approximate algorithm, MC-SAT, is presented that approximates the entire distribution.

Most Probable Explanation. MPE inference answers the question to the most probable state of the world Y given some evidence X [**?**], i.e.

$$\begin{aligned}\arg\max_{y \in Y} P(Y|X) &= \arg\max \frac{1}{Z_x} \exp(\sum_i w_i n_i(x,y)) \\ &= \arg\max \sum_i (w_i n_i(x_i, y_i)) \end{aligned} \tag{2.7}$$

Equation 2.7 follows directly from the definition of the probability distribution of the MLN provided in Equation (2.6). The 'Z' term can be left out since it is constant and therefore does not change the '$\arg\max$' operation. Moreover, due to its monotonic nature the exponential operation can be left out as well and the MPE inference reduces to maximizing the sum of satisfied formulas. In [**?**], it has been shown how to interpret ML as a modelling language for WCSPs and therefore how to transform the ML formulas into a WCSP. This makes it possible to use optimized WCSP solvers like Toulbar2[2].

First the WCSP needs to be formally defined:

Definition 2.4.2 *A Weighted Constraint Satisfaction Problem (WCSP) is a tuple* $\mathscr{R} = \langle \mathbf{Y}, \mathbf{D}, \mathbf{C} \rangle$*:*

1. $\mathbf{Y} = \{Y_1, \ldots, Y_n\}$ *is a set of n variables*
2. $\mathbf{D} = \{D_1, \ldots, D_n\}$ *is the collection of the domains of the variables in* $\mathbf{Y}$*, such that* $D_i = dom(Y_i)$ *is the domain of* Y_i*. For a given variable* Y_i *the domain may be denoted by* D_y*.* D_s *is used to denote the Cartesian product* $\prod_{Y_i \in S} D_i$ *for some subset of the variables* $S \subseteq \mathbf{Y}$*.* $\mathscr{Y} := D_Y$ *denotes the Cartesian product of all domains and hence represents the set of possible variable assignments.*
3. $\mathbf{C} = \{c_1, \ldots, c_r\}$ *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* $\mathbf{Y}$ *(V is called the scope of the constraint) such that* c_i *maps assignments (of values to the variables in V) to cost values* $c_i : D_V \rightarrow \{0, \ldots, \top\}$*. If an assignment is mapped to* $\top$*, it is considered inconsistent.*
4. *A solution to* $\mathscr{R}$ *is a consistent assignment to all variables. An optimal solution* $y \in \mathscr{Y}$ *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*).*

[2]http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/ToolBarIntro

The weights of an MLN belong to features and equivalently the costs of a WCSP belong to constraints. Henceforth, a transformation from the weights of features that can also be negative, to the costs of a WCSP, that must be positive, is needed[3]. For the successful transformation, it can be exploited that the MLN keeps its semantics when a formula is negated in conjunction with the negation of the weight of the respective formula. For the probability distribution that is derived from another probability distribution over the possible worlds of an MLN by negating one formula and its weight, it can be shown that the distributions are equal [**?**]:

$$\begin{aligned}
P_{M_{L',C}}(X=x) &= \frac{\exp\left(\sum_{i,i\neq k} w_i \cdot n_i(x) - w_k \cdot (N_k - n_k(x))\right)}{\sum_{x'\in\mathcal{X}} \exp\left(\sum_{i,i\neq k} w_i \cdot n_i(x') - w_k \cdot (N_k - n_k(x'))\right)} \\
&= \frac{\exp\left(\sum_i w_i \cdot n_i(x) - w_k \cdot N_k\right)}{\sum_{x'\in\mathcal{X}} \exp\left(\sum_i w_i \cdot n_i(x') - w_k \cdot N_k\right)} \\
&= \frac{\exp\left(\sum_i w_i \cdot n_i(x)\right)}{\sum_{x'\in\mathcal{X}} \exp\left(\sum_i w_i \cdot n_i(x')\right)} \cdot \frac{\exp(-w_k \cdot N_k)}{(\exp - w_k \cdot N_k)} = P_{M_{L,C}}(X=x)
\end{aligned}$$

where N_k is the total number of groundings(‘*True*’ and ‘*False*’) for the k-th formula.

Since in this form all weight are $w_i \in \mathbb{R}_0^+$ there is a need to scale them to match natural numbers. This operation will change the probability distribution but it keeps the set of most probable states intact [**?**].

In [**?**] it is shown that there is a non injective mapping from the grounded predicates of the MLN to the variables of a WCSP. Moreover, there is a bijective mapping from the states of the grounded MLN to the states of the WCSP. In addition each feature $\widehat{f_i}$ of the grounded MLN can be interpreted as a constraint c_i [**?**]

$$\begin{aligned}
c_i &= \begin{cases} \widehat{w}: & \text{if } \widehat{f_i}(x) = 0 \\ 0: & \text{if } \widehat{f}(x) = 1 \end{cases} \\
&= (1 - \widehat{f_i}(x)) \cdot \widehat{w} \qquad (2.8)
\end{aligned}$$

[3]This is implementation specific for the solver used in this work and in general this is not necessarily the case

The constraint c_i is defined over a set of variable assignments. Equation (2.8) shows that in case that a grounded formula from the MLN cannot be resolved to *True* then the corresponding weight can be interpreted as a cost in the WCSP. After the transformation is complete, efficient solvers for WCSPs like the Toulbar2 solver can be used.

Example. In this section, a simple example is presented that shows the transformation. For the topic of this work one might want to capture the intuition that if two words do not resemble the same real world objects that they should not be in a *coreference* relationship. This intuition can be captured with the Formula (2.9). The weight is just some arbitrary negative weight and the exact value is not important to show the principle of the transformation.

$$-\log 2 \quad \textit{coreference}(w_1, w_2) \land \textit{isGrounded}(w_1, i) \land \neg\textit{isGrounded}(w_2, i) \tag{2.9}$$

With the predicate definitions

$$\begin{array}{l} \textit{isGrounded(word, object!)} \\ \textit{coreference(word, word)} \end{array}$$

that are defined over the domains *word* $\in$ {*Cup,Mug*} and *object* $\in$ {*Mug1, Cup1*}.

First, negating the formula and its weight yields Formula (2.10). To get the set **Y** of WCSP variables all predicates are grounded and the outcome of this operation is provided in Table 2.2. Since for each grounded formula there is one feature and features correspond to constraints, the grounded formulas of the MLN are provided in Table 2.3.

$$\log 2 \quad \neg\textit{coreference}(w_1, w_2) \lor \neg\textit{isGrounded}(w_1, i) \lor \textit{isGrounded}(w_2, i) \tag{2.10}$$

The domains for the WCSP variables are defined to be $dom(Y_i) = \{0, 1\}$ that represent the truth values of the corresponding ground predicates. Each grounded MLN formula

Y_i	Grounded Predicate
Y_0	*isGrounded(Cup, Mug1)*
Y_1	*isGrounded(Cup, Cup1*
Y_2	*isGrounded(Mug, Mug1)*
Y_3	*isGrounded(Mug, Cup1*
Y_4	*coreference(Mug, Mug)*
Y_5	*coreference(Mug, Cup)*
Y_6	*coreference(Cup, Mug)*
Y_7	*coreference(Cup, Cup)*

Table 2.2.: *WCSP variables and the corresponding grounded MLN predicates*

$\log 2$	$\neg$ *coreference(Cup,Cup)*	$\vee \neg$ *isGrounded(Cup,Cup1)*	$\vee$ *isGrounded(Cup,Cup1)*
$\log 2$	$\neg$ *coreference(Cup,Mug)*	$\vee \neg$ *isGrounded(Cup,Cup1)*	$\vee$ *isGrounded(Mug,Cup1)*
$\log 2$	$\neg$ *coreference(Mug,Cup)*	$\vee \neg$ *isGrounded(Mug,Cup1)*	$\vee$ *isGrounded(Cup,Cup1)*
$\log 2$	$\neg$ *coreference(Mug,Mug)*	$\vee \neg$ *isGrounded(Mug,Cup1)*	$\vee$ *isGrounded(Mug,Cup1)*
$\log 2$	$\neg$ *coreference(Cup,Cup)*	$\vee \neg$ *isGrounded(Cup,Mug1)*	$\vee$ *isGrounded(Cup,Mug1)*
$\log 2$	$\neg$ *coreference(Cup,Mug)*	$\vee \neg$ *isGrounded(Cup,Mug1)*	$\vee$ *isGrounded(Mug,Mug1)*
$\log 2$	$\neg$ *coreference(Mug,Cup)*	$\vee \neg$ *isGrounded(Mug,Mug1)*	$\vee$ *isGrounded(Cup,Mug1)*
$\log 2$	$\neg$ *coreference(Mug,Mug)*	$\vee \neg$ *isGrounded(Mug,Mug1)*	$\vee$ *isGrounded(Mug,Mug1)*

Table 2.3.: *The grounded formulas from the example*

can now be represented as a constrained with the assigned cost that is computed with Formula (2.8).

In the example so far each ground predicate corresponds exactly to one WCSP variable. The problem can be reduced in its complexity when applying the following rationale. Each '*word*' in this example must be grounded to exactly one '*object*'. Consequently, a world where a '*word*' that is grounded to two '*objects*' is inconsistent with this assumption. A predicate where this assumption needs to hold is also called a functional predicate. In the example the '*isGrounded*' is defined to be functional which is indicated by the '*!*' in the predicate definition.

Inconsistent assignments can be eliminated in a WCSP by combining the relevant grounded predicates to a single variable. In this example the variables Y_0 and Y_1 as well as Y_2 and Y_3 can be combined to two variables reducing the total variable count from $|\mathbf{Y}| = 2^2 + 2^2 = 8$ to $|\mathbf{Y}| = 2^2 + 2 = 6$. Generally speaking the number of variables for the ' *isGrounded*' predicate is reduced from 2^n variables to 2 variables with n values, linearising the problem.

The new representation can be handed over to existing solvers for WCSPs and hence inference in ML can profit for many decades of research that has been done in the field of WCSPs.

Slice Sampling. Slice sampling [**?**] is a technique to generate a Markov Chain with the characteristic that it samples very uniformly. It is additionally favoured because it works with any random distributions and there is no need for additional knowledge about the distribution. As this procedure is very simple it is not presented here in detail but mentioned since it is used in the MC-SAT algorithm described later in this chapter(p. 27).

SampleSAT. In this section, the SampleSAT algorithm is presented that uses a hybrid strategy of Simulated Annealing and the GSAT algorithm and is further used in the MC-SAT algorithm presented on page 27.

GSAT [**?**] is a very simple way of finding assignments of variables to maximize the number of satisfied Conjugate Normal Form (CNF) formulas. It starts with a random truth assignment, afterwards greedily samples variables that will maximize the num-

ber of clauses to be satisfied by a single variable flip. This of course will make other formulas *False* again and so this algorithm continues to run until all formulas are satisfied or a maximum number of iterations is reached [**?**].
A problem with this simple approach is that it can easily run into local minima and the performance of any algorithm that belongs into this category of algorithms is determined by how well it can 'escape' those plateaus [**?**].
A solution to this problem is to combine GSAT with simulated annealing. In [**?**] it is shown that using a probability p to determine whether to do a simulated annealing or a GSAT step helps to get a more uniform sampling of solutions. More specifically, in each step of the algorithm a GSAT step is performed with probability p and a simulated annealing step is performed with probability $1-p$. The temperature in the simulated annealing step is defined as $T = 0.1$ but is subject to fine tuning of the parameters. More interesting is the actual simulated annealing step: If a neighbour that is defined as a truth assignment that differs only in one variable from the current assignment, satisfies the same or more clauses than the current assignment, the algorithms makes that move. If the neighbour satisfies fewer clauses the algorithm makes that move with probability $p_m = e^{\frac{-\delta Cost}{T}}$. Otherwise the algorithm stays on the current assignment.

The SampleSAT algorithm has shown to produce very promising results in finding optimal solutions to SAT problems.

Algorithm 1 SampleSAT(InitialTruthAssignment, MaxSteps, T)

```
$A \leftarrow InitialRandomTruthAssignment$
$p \leftarrow 0.5$
$MAX \leftarrow MaxSteps$
for $i \leftarrow 1$ to $MAX$ do
    if probability $p$ is $True$ then
        do WalkSAT step
    if probability $1-p$ is $True$ then
        $n \leftarrow neighbour$
        $Cost \leftarrow$ number of decrease of satisfied clauses
        if $Cost \leq 0$ then
            $A \leftarrow n$
        if $Cost > 0$ then
            $A \leftarrow n$ with probability $p = e^{\frac{-\delta Cost}{T}}$
```

MC-SAT. In this section, the MC-SAT algorithm is introduced [**?**]. Classical Markov Chain Monte Carlo (MCMC) methods which use Gibbs sampling or other MCMC sampling methods are problematic because in the case of near deterministic constraints the Markov Chain is not ergodic any more as the transition probabilities between states become very low [**?**]. As a consequence another approach is needed and MC-SAT has shown to guarantee producing a Markov chain that satisfies ergodicity and detailed balance, even in the presence of deterministic dependencies.

The basic idea underlying MC-SAT is to use slice sampling to sample a set of auxiliary variables that are introduced for each ground clause. The resulting set M contains clauses that are currently satisfied and are also satisfied in the next step. SampleSAT [**?**] is used to sample a new state that satisfies all the states in M. The initial state is found with the help of the hard formulas in the network. Pseudo code for the algorithm is provided in Algorithm 2.

Algorithm 2 MC-SAT(clauses, weights, num_samples)

```
[[? ]]
  $x^{(0)} \leftarrow$ Satisfy(hard $clauses$)
  for $i \leftarrow 1$ to $num\_samples$ do
    $M \leftarrow \emptyset$
    for $c_k \in clauses$ satisfied by $x^{(i-1)}$ do
      With probability $1 - e^{-w_k}$ add $c_k$ to $M$
  Sample $x^{(i)} \sim SampleSAT(M)$
```

0ptLearning

Learning in ML is only feasible in terms of weight learning instead of structure learning, i.e. learning the entire formula because structure learning presents too many challenges to be handled in real applications up to now. Structure learning uses a second order ML that can naturally reason about predicates. Two approaches are presented in [**?**].

Log-Likelihood learning. Weight learning means to maximize the likelihood of the relational databases that are provided by adjusting the weights of the formulas. The original problem is to find the vector of weights $\vec{w}$ that satisfies

$$\vec{w} = \arg\max_{\vec{w}} P(\vec{w}|d), \tag{2.11}$$

given some training database d.

Exact learning is intractable and hence approximate methods need to be used. One such a approach exploits the likelihood principles that states that all the information that is needed, is contained in the likelihood. Applying the likelihood principle to Equation (2.11) yields

$$\vec{w} = \arg\max_{\vec{w}} P(d|\vec{w}) \tag{2.12}$$

An analytic computation of Equation (2.12) is in general not always possible [**?**], but the weight function is convex, i.e. there is at least one global minimum. Consequently, optimization methods can be applied that are based on gradient descent [**?**]. Those methods use the information of the first and second derivatives to find the global maximum of a function. In order to further simplify the computation the log function can be used due to its strictly increasing nature.

$$\begin{aligned} L(D=d) &= \log P(D=d) \\ &= \sum_i w_i \cdot n_i(d) - \log\left(\sum_{d' \in \mathbf{D}} \exp\left(\sum_k w_k \cdot n_k(d')\right)\right) \end{aligned} \tag{2.13}$$

$$\begin{aligned} \frac{\partial}{\partial w_i} L(D=d) &= n_i(d) - \sum_{d' \in \mathbf{D}} n_i(d') \cdot \frac{\exp\left(\sum_k w_k \cdot n_k(d')\right)}{\sum_{d'' \in \mathbf{D}} \exp\left(\sum_k w_k \cdot n_k(d'')\right)} \\ &= n_i(d) - \sum_{d' \in \mathbf{D}} n_i(d') \cdot P(D=d') \end{aligned} \tag{2.14}$$

Equation (2.13) shows the *log* of Equation (2.12) and Equation (2.14) the partial derivative for the weight vector. **D** denotes the set of all possible databases and $n_i(d)$ the number of '*True*' groundings of the i-th formula in the data d. As it is evident, solving Equation (2.14) requires to do inference over the model and this requires counting the groundings which has been shown to be #P-complete in the length of the clauses [**?**]. As an alternative to do exact learning, *Pseudo-Likelihood Learning* has been developed.

While learning the weights of formulas there are some assumptions that are made in the algorithm presented here.

1. Closed World Assumption
   The closed world assumption states that all ground atoms that are not in the database are assumed to be *False*. With this assumption the database specifies a full assignment for one world.

2. Sets of constants are known
   The assumption states that the set of constants are finite and known. It is therefore possible to enumerate all symbols and this in turn means that the FOL syntax is a mere abbreviation for a prepositional model.

3. Variables are typed
   Typed variables help to reduce model complexity as the number of possible groundings is dramatically reduced.

4. Training databases are independent
   This assumption is important since weights can be learned from multiple databases. Assuming independence between them is natural as otherwise relationships need to be considered that never happen in the domains of discourse.

Pseudo-Likelihood learning. The *Pseudo-Likelihood* is given by Equation (2.15).

$$P^*(D = d) = \prod_{k=1}^{|D|} P(D_k = d_k | MB_d(D_k)) \tag{2.15}$$

The Pseudo-Likelihood approximates $P(D = d)$ by making independence assumptions. D_k is a ground atom and d_k its truth value. $MB_d(D_k)$ is the Markov Blanket for the k-th ground atom. In [?] it is shown that the log-pseudo-likelihood is given by Equation (2.16) and the t-th component of the gradient by Equation (2.17). $\bar{n}_{i,k}$ denotes the number of true groundings for the i-th formula where the truth value of the k-th ground atom, i.e. D_k, has been inverted. F_{D_i} is the set of indices of all the formulas where at least one grounding contains D_i.

$$logP^*(D=d) = \sum_{k=1}^{N} -\log\left(1+\exp\left(\sum_{i\in F_{d_k}} w_i \cdot (\bar{n}_{i,k}(d) - n_i(d))\right)\right) \tag{2.16}$$

$$\begin{aligned} &\frac{\partial}{\partial w_i} \log P^*(D=d) = \\ &= \sum_{k=1}^{N} \big(\bar{n}_{i,k}(d) - n_i(d)\big) \cdot \left(\frac{1}{1+\exp\left(\sum_{j\in F_{D_k}} w_j \cdot (\bar{n}_{j,k}(d) - n_j(d))\right)} - 1\right) \end{aligned} \tag{2.17}$$

With the Equations (2.16) and (2.17) an optimizer can be used that often results in a good approximation. However, as [?] points out the simplification might be too strong depending on the model at hand and consequently the results need to be carefully analysed.

In order to combine multiple learning databases the assumption is made that the databases are independent which is a valid assumption as the instances in the database might only occur in the context of the database and not with the instances of others. In the case of the log-likelihood the problem is simply solved by the sum of the individual log-likelihoods. Further details can be found in [?].

Chapter 3

Coreference and Entity resolution

0ptThe Model

In this section, two approaches are presented where one is a model of coreference and the other of entity resolution. In both models the underlying predicates are mostly the same and henceforth are introduced first.

A basic assumption of this work is that a sentence given in natural language can be transformed into a logical representation that captures the semantics of the sentence in the context of instructions. Consequently, a symbolic representation is introduced where symbols represent the different entities that can be found in the instructions. Predicates are used to represent the relationships between them or assign attributes to the entities.

The evidence that is used to generate the predicates can be roughly divided into the following groups:

Syntactic relationships. Syntactic relationships are relationships between words in the text or assignments of properties to those words. First of all, each word is assigned its Part-of-Speech (POS) tag. Moreover, several syntactic dependencies between the words in a sentence can be identified. For example, in the sentence '*Put the red cup on the table.*' there is a dependency of the word '*red*' and the word '*cup*' that states that the first is the adjectival modifier of the second. Moreover, it can be identified that the '*Put*' is connected to the '*table*' via a prepositional modifier. External tools can be used to generate this evidence and it was not part of this work to come up with rules to deduce it. See Section 3.2 for implementation details.

Taxonomy relationships. Taxonomy relationships refer to the relationships that exist between the concepts of objects. For example, the concept '*container*' is in a hyponomy relationship with the concept '*cup*' since the '*cup*' is some form of '*container*'. Deriving these relationships for the objects in a text defines a taxonomy of concepts.

Semantic roles. Semantic roles refer to evidence that is created in order to identify the semantics of a word in a text in the context of the action verb in the sentence. In the previous example the '*cup*' is the object being moved by the '*put*' action and is therefore called the *theme* of the action.

Distance relationships. Distance relationships are established between each word pair to determine the sentence distance between each word. In [**?**] it has been shown that within three sentences roughly 90% of all coreference relationships can be found. The research focused on newspaper articles but in this work it is assumed that the general influence is the same in both contexts even though the exact influence might be slightly different. This suggests that the distance needs to be taken into account when inferring coreference.

Object relationships. Object relationships refer to the relationships between instances of object concepts in the real world. These could be geometric relationships or colour properties. For example, the colour of a '*cup*' or the fact that the '*cup*' is located on top of a '*kitchen table*'. This kind of knowledge also has to be captured in the knowledge base about the objects that are used.

As can be seen from the previous elaborations there are several types of entities. More specifically there are the following types that need to be considered: *Senses* refer to the sense of a word from the '*words*' domain. The sense is in a hyponomy relation with several concepts from the '*word concepts*' domain. Moreover, each word is assigned a semantic role from the '*roles*' domain. The objects that exist in the robot's current belief state of the world are '*instances*' of '*object concepts*'.

0ptA model for coreference resolution

Coreference resolution in this model assumes that no grounding information is available. Additionally, it is important to note that this model is not supposed to replace

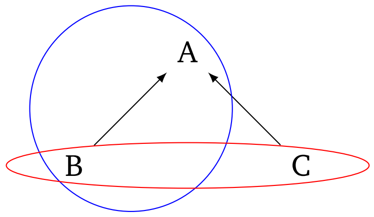


Figure 3.1.: *Concept A and B are as similar as B and C with the chosen metric.*

existing coreference systems that were developed in the NLP community but to complement them as it identifies coreference that cannot be determined by the existing systems as has been elaborated on in Section 1.2.

Several semantic features are taken into consideration when inferring coreference. The word sense and the path through the taxonomy of two words are among the most important ones. Two words are more probable to be in coreference if the semantic distance is small. If the semantic distance is zero, two concepts are exactly the same. Different similarity measures for the semantic distance exist like the WUP metric [**?**]. Despite the abundance of available metrics, in this work another approach is used. This is chosen due to the fact that every metric that gives a numeric value of similarity needs to be further discretized in the implementation that is chosen. An alternative that can be modelled very naturally in ML, counts the common edges in the taxonomy. This approach has the drawback that all concepts that have the same direct parent have the same similarity score to each other and to the parent concept. This is exemplified in Figure 3.1. The concept *A* and concept *B* are as similar as the concepts B and C. As the evaluation in Section 3.4 will show this drawback is a reasonable compromise.

A big difficulty arises in the case that not all the word senses are available or even no word sense is available at all. In this case another great feature to determine coreference is the use of semantic roles and their relationship to each other in dependency of the distance they appear in. For example, in the set of instructions:

Mix the fruits. $\boxed{\textbf{MixPlace}}$
Serve. $\boxed{\textbf{ServeTheme}}$

The ‘*theme*’ of the ‘*serve*’ action and the ’*place*’ of a ‘*mix*’, if appearing in directly consecutive tasks are very likely to be in coreference.

An additional difficulty with coreference exists when identifying individual objects and a word occurs multiple times in a set of instructions but they refer to different object entities. For example, the instruction set:

Add fruits to a bowl.
Mix. $\boxed{\textbf{MixPlace}}$
Serve in a bowl.

One interpretation of the instructions could be that the word '*bowl*' in the first sentence refers to a mixing-bowl where the ingredients are mixed. The word '*bowl*' in the third sentence on the other hand refers to an eating-bowl that is used for serving. This is difficult to handle and particularly difficult to handle only with syntactic features. Consequently, the semantic information is a key element in the correct identification of the coreference. For example: in the given example the word sense would reveal that the word '*bowl*' in the first sentence has the sense of a '*Mixing-Bowl*' and the '*bowl*' from the last sentence the sense of a '*Eating-Bowl*'. This already indicates that these two are not the same real world object. Moreover, even in the absence of the senses, the roles of the two words can indicate whether it is probable that they are in coreference or not.

0ptA model for entity resolution

In order to ground words to objects, the most important indicator is the hyponomy relationship between the sense and the *concept* of the word and the relationship of the objects and the *concepts* of the objects. If the correct sense is known as well as the object's concepts, then a semantic distance can be calculated that picks the objects in the world that is closest to the word sense.

Since the semantic distance between the concept of the word and the concept of the object is to be computed they must exist in the same taxonomy. Figure 3.2 shows an example of the word 'cheese' that has a path through the taxonomy for a specific sense. The cheese object defines the same path through the taxonomy and therefore the semantic similarity will be very high. It is important to note that any more specific concept than 'Cheese' will receive the same similarity score as has been pointed out in the previous section. This is a desirable property because if an instruction sates

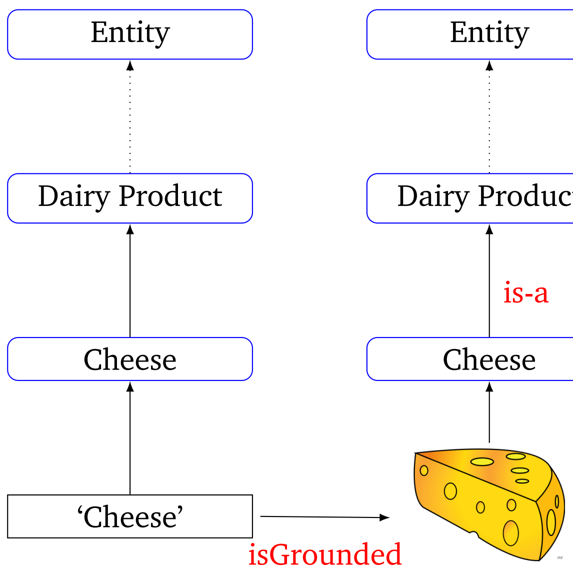


Figure 3.2.: *A word is matched to an object instance based on the taxonomy of both entities.*

to '*Put cheese on the table.*' it is not specified whether to take '*Goat Cheese*' or maybe '*Gouda*'. At this point other features could be used to select the correct cheese, like the operator's preference.

Using a taxonomy has several positive implications. First, if a concept is encountered that is not a member of the current knowledge base, then the taxonomy provides good generalization properties. This is achieved by the fact that the semantic distance will select the object that is closest to any object that exists in the real world. Of course this assumes that the concept of a word is known.

Furthermore, the situation where a specific object is referenced in the text that is not available in the world is very common, like a missing cooking utensil. It is therefore important to be able to find appropriate alternatives. Using a taxonomy and the semantic distance is a very effective way of choosing those: if there is no object in the world that is expected, the object with the closest semantic distance is chosen. For a example if an instruction expects a '*Mug*' but no 'Mug' is available then a 'glass' could be chosen that offers the same features as a 'mug', i.e. it can hold a substance. This assumes that objects that have a close semantic distance, share such core properties. This cannot always be guaranteed but is assumed in this work and will be further elaborated on in Section 3.5.

As this work has been developed in the context of the PRAC and its goal is to build a knowledge base of action specific background knowledge, it is desirable to capture the relationship between objects and their use in actions. For example, the instruction '*Put cheese on the table.*'. In the case that the robot needs to choose from multiple different '*cheeses*', it would be desirable if the robot performs a lookup in the knowledge base which cheese is usually the right choice in this situation with this kind of action. There are multiple parameters that can trigger a different selection. First, the availability of different types of cheese but also the preference of the current operator is a parameter that could be taken into consideration. Yet, this last method has not been implemented in this work but leaves room for improvement.

Additionally, if two words refer to the same object instance they are in a '*coreference*' relation and this needs to be used when grounding objects. In the grounding model it is assumed to be given the '*coreference*' evidence about the words. The '*coreference*' relation is essential when multiple instances of the same object are present and two words need to be grounded to the same instance and not just to the right class of objects. Moreover, it makes it possible to use information about one word for the grounding of another. Moreover, if the '*coreference*' relationship can be used to infer missing word senses by using the word sense of the referenced word.

On top of the features discussed so far it is important to consider other attributes of objects that set them apart in a set of objects of the same concept. For example, in order to be able to pick the 'cup' on the 'table' the geometric relationship between the objects needs to be taken into account. Moreover, the attributes of objects are of interest as well. For example, in order to be able to picks the 'red cup' the knowledge base needs to include this colour information about the object. In the scope of this work not a complete list of possible relationships between objects and attributes of them can be taken into account but only a small subset. However, this is enough to show the applicability of this approach and the underlying principles.
Syntactic dependencies can provide additional evidence about the relationships that are to hold between the objects. For example, in the instruction

'*Put the red cup on the table.*'
'*Grasp the green cup on the table.*'

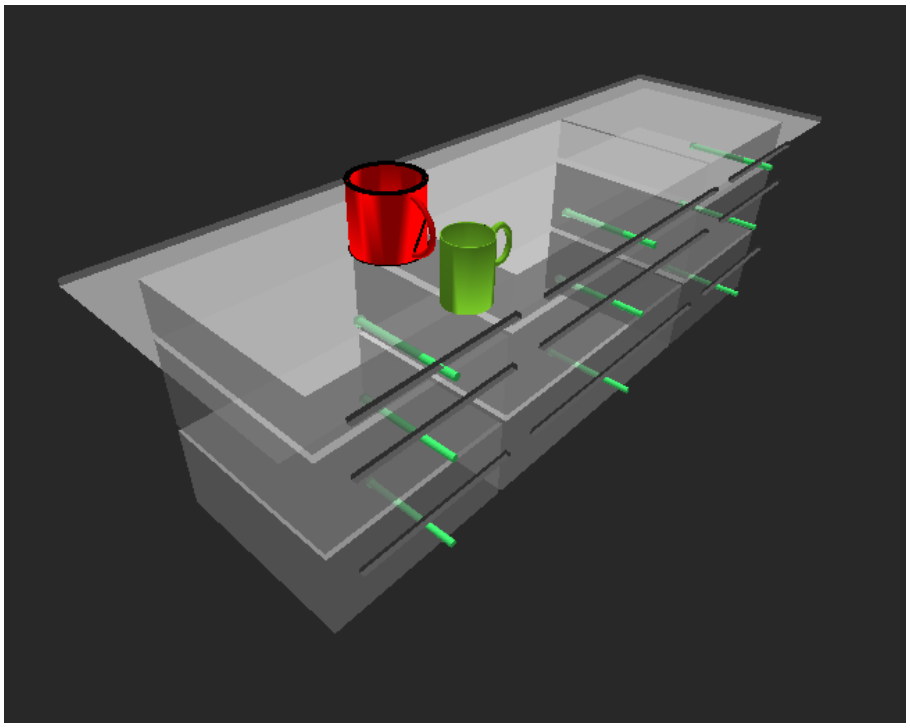


Figure 3.3.: *Two drinking mugs exist in the world.*

The '*red*' is an adjectival modifier of the noun '*cup*'. This additional evidence needs to be exploited in order to be able to pick a '*red cup*' in the world that is not already on the referenced table. Contrary, in the second instruction the '*green cup*' needs to be chosen. The problem is visualized in Figure 3.3. Furthermore, the roles of the entities in the text provide information about the relationships between the objects. In the example the roles infer that the theme of a '*put*' action should not be located on the goal of that action. This information is very valuable in case the word senses are not present. Again, contrary in the second instruction, the '*grasp*' action requires the '*cup*' to be located on the '*table*'. However, all of this requires to have a very good knowledge about the objects in the world.

To find a suitable model for entity resolution, instruction dynamics pose an especially difficult modelling problem. The objects that appear in the text do not necessarily exist in the belief state of the robot at the point of text processing. Especially objects that are created by manipulation of the environment. For example, '*dough*' that is a mix of different ingredients cannot be located in the world at the time of instruction processing. Those words, virtual or actual words, are grounded to some unnamed virtual object instance. This is helpful because this could trigger the perception system of the robot as a new object is expected that needs to be located and identified. The knowledge base could be updated with this information and processing could continue. More on this problem can be found in the model evaluation in Section 3.5.

0ptA joint model

A joint model is combining coreference and entity resolution into one model. The idea is that the information of one model is giving evidence to the other and vice versa, such that overall prediction quality increases. One example is already used in the model for the grounding in the fact that words that are in a *coreference* relationship ground to the same objects. The feature could be used the other way around as well stating that two items are in a coreference relationship exactly if they ground to the same object.

In order to accomplish this, it will be shown that the two models can simply be combined into one model by joining the sets of rules.

0ptImplementation

In the following section an overview of the core components of the implementation of the coreference and entity resolution model is provided. First, a brief overview over the entire processing pipeline for natural language is given. Afterwards, the different components that are used for the implementation are presented. Furthermore, the details of the implementation of the ML formulas are provided.

0ptNatural Language Processing

The entire system design is provided in Figure 3.4 and shows the pipeline of processing steps. First, the system has to disambiguate what the instruction generally refers to. The next step is to download a set of instructions from the internet and make it available for further processing[1] [**?**]. Step 3 performs syntactic parsing using the Stanford

[1]e.g. from wikihow.com or ehow.com

parser [**?**], a probabilistic first order grammar that reveals all syntactic relationships between the words and the syntactic features of the words, like the part of speech tag. The fourth step involves identifying which action verbs are used in which sentence. In step 5, word sense disambiguation and role labelling in the context of PRAC [**?**] is performed as elaborated on in Section 2.1. The next step is the topic of this work. The collected evidence produced by steps 1-5 is passed into the next component that identifies the coreference relationships and grounds the objects identified in the text into the perceived belief state of the robot. In the last step the instructions are passed into the execution engine of the robot.

0ptModel implementation

In a first step the provided text is transformed into a symbolic representation. This is achieved by introducing symbols for each word so that the word symbol identifies the word in the text. The encoding is as follows:

«WORD»_«ID»_S_«SentenceID»

The word-ID is chosen based on the parse tree that is generated by the Stanford parser and identifies the word in its sentence. The sentences are numbered in ascending order. For example, the word symbol '*Pasta_1_S_0*' means that this symbol represents the word '*pasta*' with the id '*1*' in the parse tree in the first sentence of the text.

WordNet [**?**] is used as the taxonomy of the words. It is a lexical database of English and words are grouped into sets of cognitive synonyms(synsets), where each set represents a distinct concept. This means that words are grouped by their semantic meaning. Between the synsets exist relationships that span a taxonomy of concepts. Naturally, this leads to different taxonomies for verbs and nouns. In the following only the taxonomy for nouns is used as identifying the correct verb sense is part of another project [**?**]. The relationship between synsets that is used in the context of this work is hyponomy (also called '*is-a*' relation). The hyponomy relation is exactly the one described in Section 3.1. Access to WordNet can easily be gained using the NLTK[2] toolbox for the Python[3] programming language.

[2]http://nltk.org/
[3]http://www.python.org/

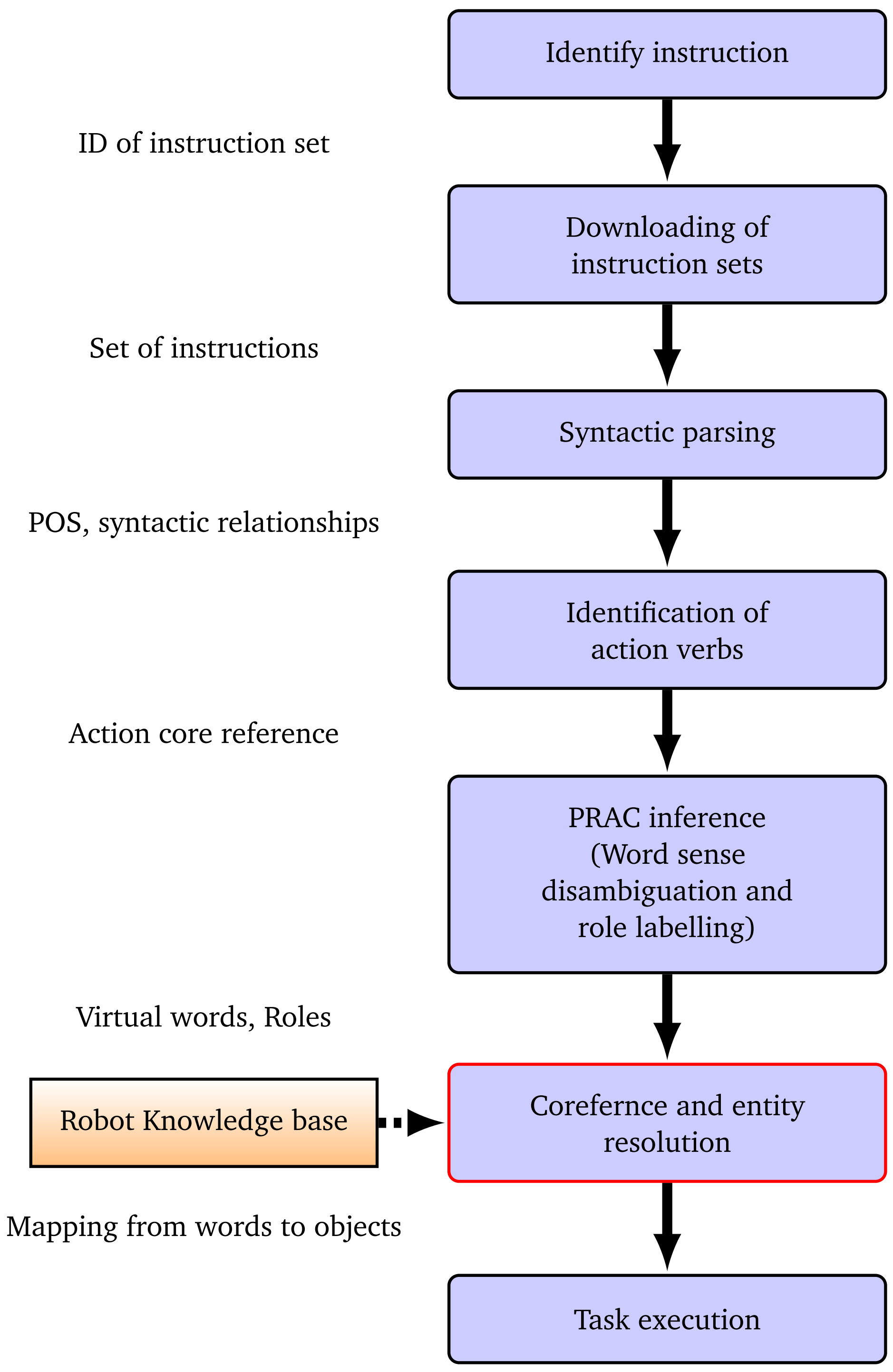

Identify instruction
ID of instruction set
Downloading of instruction sets
Set of instructions
Syntactic parsing
POS, syntactic relationships
Identification of action verbs
Action core reference
PRAC inference (Word sense disambiguation and role labelling)
Virtual words, Roles
Robot Knowledge base
Corefernce and entity resolution
Mapping from words to objects
Task execution

Figure 3.4.: *System design for instruction processing. The arrow specifies what is added in information in each step. Coreference and entity resolution(red box) is the topic of this work.*

The system is implemented with Python and as a Robot Operating System (ROS) node[4]. ROS is a middleware system to run on autonomous robots. ROS makes inter process communication over the network easy. In this work it is mainly used to communicate with the robot's belief state of the world.
To get a list of objects in the world that are available in the belief state of the robot, the KnowRob [**?**] knowledge base is used that has been developed by Moritz Tenorth[5]. KnowRob is essentially an ontology that stores all high level knowledge of the robot. This includes information about the objects in the world and the environment but also about the actions taken by the robot. KnowRob stores relationships between objects, like geometric relationships between objects and object properties, like the colour. The layout of the upper levels of the ontology is inspired by the OpenCyc [**?**] ontology that is used to store general knowledge. In the context of this work only the ability to extract information about the objects in the environment is used. Despite KnowRob's own taxonomy, in the context of this work a direct mapping from KnowRob concepts to WordNet concepts has been created. Ontology mapping is an entire new research area and out of the scope of this work. The mapping consists of 54 lower level concepts. Those concepts are mostly part of the training database that has been used for the experiments. The creation of a mapping for other concepts is a work in progress. Moreover, a list of relationships that are taken into consideration needs to be provided by the user(e.g. on(object,object) or in(object,object)). This has the benefit that the system can be configured quite easily to new requirements.

The roles that are used in the context of PRAC are inspired by the Berkeley FrameNet [**?**][6] corpus. FrameNet is a lexical database based on *frame semantics*. The sense of a word invokes a frame that describes the situation in which the sense occurs in. For example, the word '*to put*' in the context of '*To put something somewhere*' invokes the frame '*placing something somewhere*'. It is important to understand that multiple words can trigger the same frame. Within a frame, the different words of the sentence get semantic roles assigned. In the example frame the word '*something*' has the role '*Theme*' of the frame. In this work the roles that are used by the FrameNet form the basis for the role labelling.

[4]http://www.ros.org/wiki/
[5]http://ai.uni-bremen.de/team/moritz_tenorth
[6]https://framenet.icsi.berkeley.edu/fndrupal/home

As the modelling language for the system ML is used as described in Section 2.4. The ProbCog[7] toolbox by Dominik Jain[8] is used for inference and learning as it implements the algorithms described in Section 2.4.3 and in Section 2.4.4.

In order to implement the model from Section 3.1, rules for ML need to be defined that capture the intuitions that are described above. In the ProbCog modelling language, predicates are written in lower case letters. Variables appear as parameters of predicates and are also written lower case. Constants start with capital letters. The modelling language used in the ProbCog toolbox additionally allows to use some abbreviations when defining formulas: a '*+*' before a variable name indicates that this variable will be expanded over the entire domain. Hence, for each domain element a new formula will be created for which the weight can be assigned or learned. For example, for the rule

hasPOS(w,+pos)

with the domain $pos \in \{NN, VB\}$, the following formulas are created

hasPOS(w, NN)
hasPOS(w, VB)

Functional predicates are indicated with the '*!*' operator after the variable name. For the predicate definition

isGrounded(word,instance!)

the word functionally determines the instance to which it is grounded. In other words the word needs to ground to exactly one instance.

For the functional grounding predicate '*isGrounded*' a domain element, '*NULL*', is introduced. This is needed since functional predicates require that exactly one grounding is '*True*' for the predicate. This does not always make sense but the computational benefits of functional relationships should be exploited. Hence, if the predicate cannot

[7]https://github.com/opcode81/ProbCog
[8]http://ias.in.tum.de/people/jain

find a domain member to be '*True*' then it assigns the value '*NULL*'. For example for the grounding predicate '*isGrounded(word,instance!)*', if no instance for the word can be found in the world then the word is grounded to the '*NULL*' symbol.

Syntactic relationships consist of two types of predicates: the part of speech tag, '*hasPOS(word,pos!)*', and syntactic dependencies between words. For example, '*amod(word,word)*' which states that the second word is an adjectival modifier of the first. There are several more syntactic dependencies that the Stanford parser generates but for brevity only one is presented here[9]. '*hasPOS(word,pos!)*' assigns each word a part of speech tag. For example the relationship '*hasPOS(Pasta_1_S_0, NN)*' states that the word 'pasta' has the part of speech '*Noun*'.

Taxonomic relationships induce the taxonomy of the concepts of word senses. The predicate '*hasSense(word,sense!)*' assigns each word exactly one sense ID. The sense itself defines a path in the taxonomy. The '*isaW(sense,concept)*' predicate provides a relationship between the sense of the word and the concepts in the taxonomy. For example, the word '*Shaker*', in the sense of a cocktail shaker has the following block in the resulting MLN:

hasSense(Shaker_10_S_0, Shaker_n_03)
isaW(Shaker_n_03, Shaker_n_03)
isaW(Shaker_n_03, Physical_Entity_n_01)
isaW(Shaker_n_03, Artifact_n_01)
isaW(Shaker_n_03, Entity_n_01)
isaW(Shaker_n_03, Object_n_01)
isaW(Shaker_n_03, Instrumentality_n_03)
isaW(Shaker_n_03, Container_n_01)
isaW(Shaker_n_03, Whole_n_02)

Semantic roles identify the role a word has in the context of an action. For this work only roles that identify the roles of objects are needed and consequently all other roles are ignored. The predicate '*hasRole(word,role!)*' is used to state that a word has a certain role. For example, in the sentence:

Add ice to the glass.

[9]for a complete list, see: http://nlp.stanford.edu/software/dependencies_manual.pdf

The word ‘*glass*’ is the goal of the ‘*Add*’ action. Therefore the predicate ‘*hasRole(Glass_5_S_0, AddGoal)*’ is ‘*True*’. It is important to note that the name of the role encodes the verb that the role belongs to. For example, for the role ‘*AddGoal*’, where it is clear from the role name that the role defines the ‘*Goal*’ of an ’*Add*’ action. This has strong implications and is further discussed in Section 3.5.

Sentence distance relationships are established between each word in the text. Four different predicates are created where each predicate represents a distance. For example, the predicate ‘*distance0(word,word)*’ states that, if true, the two words are in the same sentence. There are distances from ‘*0-3*’ and one for distances greater than ‘*3*’.

Object relationships capture knowledge about the objects present in the robot’s belief state of the world. In order to be able to ground objects into this perceived world there needs to be a ML representation of it. This is accomplished by having instances of object concepts. Various relationships can then be represented by predicates that are defined over two object instances. For a refrigerator the MLN representation looks as follows:

isInstanceOf(Refrigerator1, Refrigerator)
isaI(Refrigerator, Appliance_n_2)
isaI(Refrigerator, Physical_entity_n_01)
isaI(Refrigerator, Refrigerator_n_01)
isaI(Refrigerator, Home_appliance_n_01)
isaI(Refrigerator, Artifact_n_01)
isaI(Refrigerator, Entity_n_01)
isaI(Refrigerator, Consumer_goods_n_01)
isaI(Refrigerator, Object_n_01)
isaI(Refrigerator, Durables_n_01)
isaI(Refrigerator, White_goods_n_01)
isaI(Refrigerator, Commodity_n_01)
isaI(Refrigerator, Whole_n_02)

‘*isInstanceOf(instance, conceptID)*’ instantiates a concepts.
The ‘*isaI(conceptID, object_concept)*’ relationship is the equivalent of the ‘*isaW(word_sense, word_concept)*’ relationship in the object taxonomy. The taxonomy

of the object concepts is independent of the taxonomy for word concepts. From a knowledge engineering point of view they are two separate independent taxonomies. However, in this work it is decided that both use the WordNet taxonomy. Nevertheless, two different relationships are created to be able to have the mentioned distinction. Geometric relationships are relationships between the instances which indicate some geometric relationship between two instances. For example the relationship '*on(instance, instance)*' states that the object that is represented by the first instance is physically located on top of the other instance.

The query predicates used are '*coreference(word,word)*' and '*isGrounded(word,instance)*'. '*coreference*' is '*True*' if two words represent the same real world object. '*isGrounded*' is '*True*' if a word represents an object in the perceived world of the robot as explained in Section 1.2.

0ptPreprocessing

When processing a text, many parts of the texts are not absolutely necessary for the inference. Despite the fact that learning from negative examples adds information to the model, the model complexity needs to be reduced in order to be able to learn the models in proper time. The following describes some optimizations that were applied.

Verbs. Verbs are provided with a sense, the taxonomy of the sense and the part of speech tag. However, the verb itself is not necessary for the resolution of the objects that are contained in the text since the roles have the action verb encoded in their name. Consequently, no connection needs to be made between a role and its verb. If the verb is needed as part of any syntactic dependency then the sense of the verb is set to "NULL" in order to further reduce the model size.

Stop words. Stop words in this context are words that have no semantic meaning for the action and are not needed in any syntactic dependency.

Syntactic Dependencies. Some syntactic dependencies can be deleted from the evidence. The Stanford parser produces many dependencies that are not used in any formula and as a consequence are not needed.

0ptMarkov Logic Formulas

The predicates presented in Section 3.2.2 are used to create formulas that capture the information that is needed to resolve the '*coreference*' and '*isGrounded*' query predicates. While designing the formulas it is important that the total number of formulas does not grow too large as this will slow down learning. Additionally, the number of groundings of a formula needs to be kept low to reduce the needed amount of memory. As a result the formulas presented are always a trade-off between expressiveness and performance and should be evaluated as such.

0ptFormulas for coreference resolution

The '*coreference*' relation as it has been described earlier has the transitivity property which can be implemented as a hard formula, i.e. a formula that always needs to hold.

$$\mathit{coreference}(w_1,w_3) \Rightarrow (\mathit{coreference}(w_1,w_2) \Leftrightarrow \mathit{coreference}(w_2,w_3))$$

Moreover, certain combinations of part of speech tags usually are not in a *coreference* relationship, e.g. a noun with an adjective. As a result, the following formula assigns a weight to each possible combination

$$\mathit{coreference}(w_1,\ w_2) \land \mathit{hasPOS}(w_1,+pos_1) \land \mathit{hasPOS}(w_2,\ +pos_2)$$

If a word is assigned the sense '*NULL*', then it is assumed that this word cannot be in any coreference relationship. This does not include the case where the sense of a word is unknown as for that case the '*hasSense*' is defined to be '*NONE*'. Learning the weight of the following formula leads to a large negative weight. This makes worlds where '*coreference*' between two words is '*True*', less probable if one of the words has the sense '*NULL*'.

$$\mathit{hasSense}(w_1,NULL) \Rightarrow \mathit{coreference}(w_1,w_2)$$

Coreference between two words can be inferred with the help of the roles, senses and the distance between the words. This relationship is captured in the next formula.

$$hasRole(w_1, +r_1) \wedge hasRole(w_2, +r_2) \wedge coreference(w_1,w_2) \wedge distance(w_1,w_2,+d)$$

This formula expands over the roles of the words and consequently making the roles a part of the model. As a result, the model size increases quadratically in the number of roles.

0ptFormulas for entity resolution

Grounding words to objects that reside in the robot's belief state requires to have knowledge about the objects that exist in this belief state. This work assumes that there exists a mapping from objects in the world to WordNet concepts and hence the '*isaI(«Object_Concept_ID», «Object_Concept»)*' predicate is defined over WordNet concepts. This allows to semantically compare the sense of a word with an object in the world.

$$isGrounded(w,i) \wedge hasSense(w,sid) \wedge isaW(sid,c) \wedge isInstanceOf(i,cid) \wedge isaI(cid,c)$$

This formula is '*True*' whenever the real world item and the word have a concept in common. This is particular powerful since this formula alone can be interpreted as a semantic distance between two items in the taxonomy. When the number of true groundings is large, i.e. the number of common edges in the taxonomy is large, then the world is more probable since each time the weight of the formula adds to the overall weight of the world. Of course this assumes the weight to be positive.

Moreover, it is evident that the part of speech tag of a word is important for the grounding. Usually only nouns are grounded to objects and as a result the next formula captures the relationship of the part of speech tag with the grounding to the '*NULL*' entity.

$$hasPOS(w, +pos) \Rightarrow isGrounded(w, NULL)$$

Additionally, it is intuitive that words where no sense can be allocated cannot be grounded with this model since no semantic information is available and therefore the basis for a grounding decision is missing. The resulting formula is shown below.

$$\mathit{hasSense(w, NULL)} \Rightarrow \mathit{isGrounded(w, NULL)}$$

In order to capture knowledge about the objects in the belief state, two more relationships are introduced. '*on(instance, instance)*' and '*in(instance, instance)*'. To make a grounding based on these properties two additionally formulas are to be introduced.

$$\mathit{hasRole}(w_1,+r_1) \wedge \mathit{hasRole}(w_2,+r_2) \wedge \mathit{isGrounded}(w_1,i_1) \wedge \mathit{isGrounded}(w_2,i_2) \wedge \mathit{on}(i_1,i_2)$$

and

$$\mathit{hasRole}(w_1,+r_1) \wedge \mathit{hasRole}(w_2,+r_2) \wedge \mathit{isGrounded}(w_1,i_1) \wedge \mathit{isGrounded}(w_2,i_2) \wedge \mathit{in}(i_1,i_2)$$

To learn the weights of the formulas a training database has to be created that contains the annotated data. With the training database the weights of the formulas are learned in a supervised fashion and the model can then be used for inference. This has the benefit that the engineer does not need to know the extent to which the formulas hold, hence he does not need to know the hardness of the constraint. This is determined by the learning algorithms based on the annotated training data. With the described formulas several experiments have been conducted which are presented in the next section.

0ptExperiments

The following experiments are all carried out in a kitchen environment that is inspired by the kitchen which is assembled in the lab of the IAS group. An accurate model of the kitchen is available for the KnowRob software containing all elements that are used in the kitchen and hence known to the robot. Figure 3.5 shows the empty kitchen model without a robot and kitchen utensils.

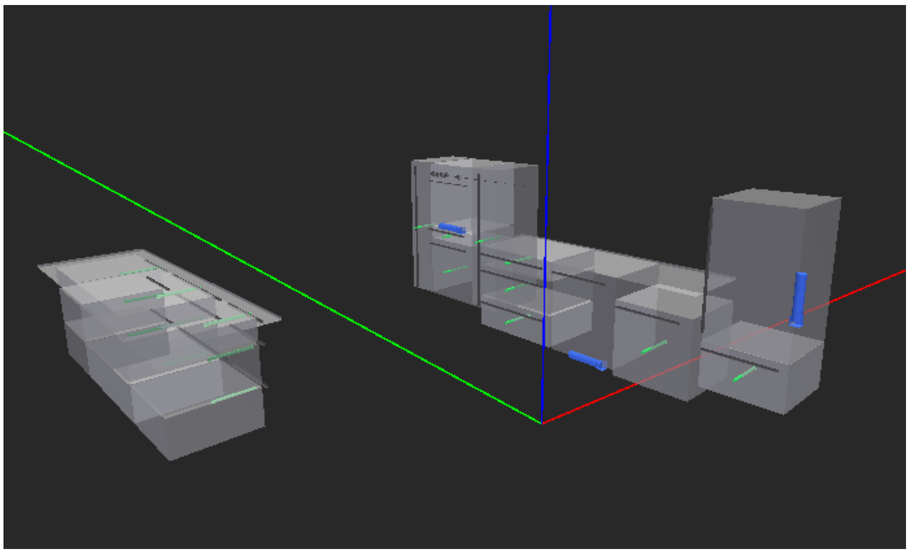


Figure 3.5.: *The empty kitchen model from KnowRob.*

Since the full model of the kitchen is very complex, a reduced version is used for the experiments. The training data only consists of a '*Kitchen-table*' and a '*Refrigerator*' where the objects mentioned can be stored, i.e. they are in a geometric relationship with those objects. Each object that is needed for a text is at least once appearing in the kitchen environment. Most of the time multiple instances with different properties are present to force non trivial grounding decisions.

For the experiments, multiple training databases are needed. Samples from the ehow corpus are not a feasible option at this point for a couple of reason. First, the length of the sets is too long to be processed by the current implementation. The average length of the 8783 recipes is approximately 173[10]. Additionally, the different recipes include many verbs for which no models exists at this point. A more detailed discussion of the limitation of the current model is provided in Section 3.5. Consequently, for the experiments, a total of 20 short texts, containing a total of 57 sentences, have been manually annotated by the author. Each set of instructions contains an average of 12 words. Instruction sets that are this short have to be chosen due to the complexity the words introduce in the model. The texts can be found in the Appendix A. The texts contain the verbs: '*to put*', '*to add*', '*to mix*', '*to fill*' and '*to serve*' and each sentence only has one verb. Each text contains of a new set of objects that do not exist in the other texts. In all texts a total of 67 different objects are named. For inference all predicates

[10]based on the simplification that words are always separated by a " "(space)

that are part of the evidence are assumed to fulfil the closed world assumption, i.e. if they do not appear in the evidence, they are assumed to be '*False*'.

Different metrics are to be taken into account when evaluating how well the different models perform. To find appropriate ways to measure the performance of the models it is necessary to keep the goal in mind: To select the correct objects for the right words. In this case, precision, recall as well as the F1 score are useful metrics.

'*Precision*' indicates how accurately the model works. Precision is the fraction of correctly identified results of all the results. It is computed as provided in Equation (3.1).

$$precision = \frac{tp}{tp + fp} \tag{3.1}$$

'*tp*' are the number of '*true positives*'. '*fp*' are the false positives.

The recall measures the fraction of relevant predictions made. Is is computed as provided in Equation (3.2).

$$recall = \frac{tp}{tp + fn} \tag{3.2}$$

Usually, if the recall increases, then the precision decreases and if the precision increases the recall decreases. This in general makes it necessary to trim the model used in order to get a good trade-off between recall and precision.

The F1-score combines precision and recall and gives the harmonic mean of the two. It is computed from precision and recall as given in Equation (3.3).

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3.3}$$

Although the three metrics just introduced are standard machine learning evaluation methods, in case of functional predicates not all three need to be computed. This is due to the fact that precision and recall are equal for functional query predicates. This is easy to see: For a result predicate '*isGrounded(Word1, Instance1)*' which is wrongly grounded to '*False*' a false-negative is counted. Since the predicate is functional it is necessary that exactly one other grounding of the predicate is found to be '*True*'. Hence creating a false-positive count. On the other hand, if the predicate is wrongly grounded to '*True*', then a false-positive is counted. This will create an additional false-negative since this is the only possible grounding for the actual object. This leads to the result that the counts of false-positive and false-negatives have equal numbers. As a result, since precision and recall only distinguish in these two counts the metrics lead to the same results. Ergo, the F1 score has the same value as well and one of the metrics needs to be sufficient.

Each experiment follows the same structure: The set of annotated data is randomly split into a training set and a query set. The set sizes vary in the different experiments. For each query the precision, recall and F1 score are computed. This procedure is repeated 10 times and an overall average is computed. To get statistically relevant data this entire procedure is again repeated 10 times and the average scores are computed. As a result, for each experiment, a total of 100 models are learned and the number of queries varies according to the set sizes.

Subsequent to the description of the conducted experiments and the presentation of the results possible interpretations are provided in Section 3.4. A critical analysis of the chosen model is provided in Section 3.5.

OptCoreference experiments

For the coreference model two sets of experiments are conducted.

Experiment 1. In the first experiment, all word senses are provided in the query databases and no grounding information. The database is split in sets of different sizes. The results are provided in Table 3.1a. As to be expected the results show that the run with 18 training databases receives the highest scores for recall as well as precision and the lowest for the run with only two training databases. Figure 3.6 visualizes the

∅# of formulas	# Queries	# Training DBs	∅F1	∅ Precision	∅ Recall
1635	2	18	0.793	0.774	0.865
1615	10	10	0.727	0.737	0.814
1183	15	5	0.660	0.680	0.772
1021	17	3	0.640	0.617	0.826
816	18	2	0.592	0.569	0.799

(a) Experiments for coreference where all senses are provided as evidence.

∅# of formulas	# Queries	# Training DBs	∅F1	∅ Precision	∅ Recall
1635	2	18	0.690	0.617	0.851
1615	10	10	0.600	0.522	0.762
1183	15	5	0.558	0.473	0.782
1021	17	3	0.513	0.428	0.778
816	18	2	0.442	0.363	0.752

(b) Results if symmetric results are filtered out of the result set for the first experiment

Table 3.1.: *Results for the first experiment, including all word senses*

learning curves for precision and recall. It is interesting to observe that the difference between the highest and the lowest value for recall is only 0.09 points whereas the difference for precision is 0.205. Especially interesting is the large increase of the precision in the runs from 2 to 10 training databases which indicates good generalization properties while keeping the recall nearly constant.

The '*coreference*' relationship is defined to be symmetric and consequently each object is in relation with itself. The author considers these symmetric relationships as easy to predict and consequently the scores in Table 3.1a do not express the quality of the model in the best possible manner. Table 3.1b shows the scores for the same experiment without the symmetric predictions. The table shows that especially the precision is a lot lower with this reduced look at the results. Recall is still very high but decreases a bit stronger with less training databases.

Experiment 2. As a second experiment it is tested how well the model performs when no word senses are provided as evidence in the query databases. The prediction then mainly relies on the relationship of the roles in the text. The same setting as in the previous experiment is applied and the results are provided in Table 3.2a. As in the previous experiment the recall is nearly constant as can been seen in Figure 3.7. The

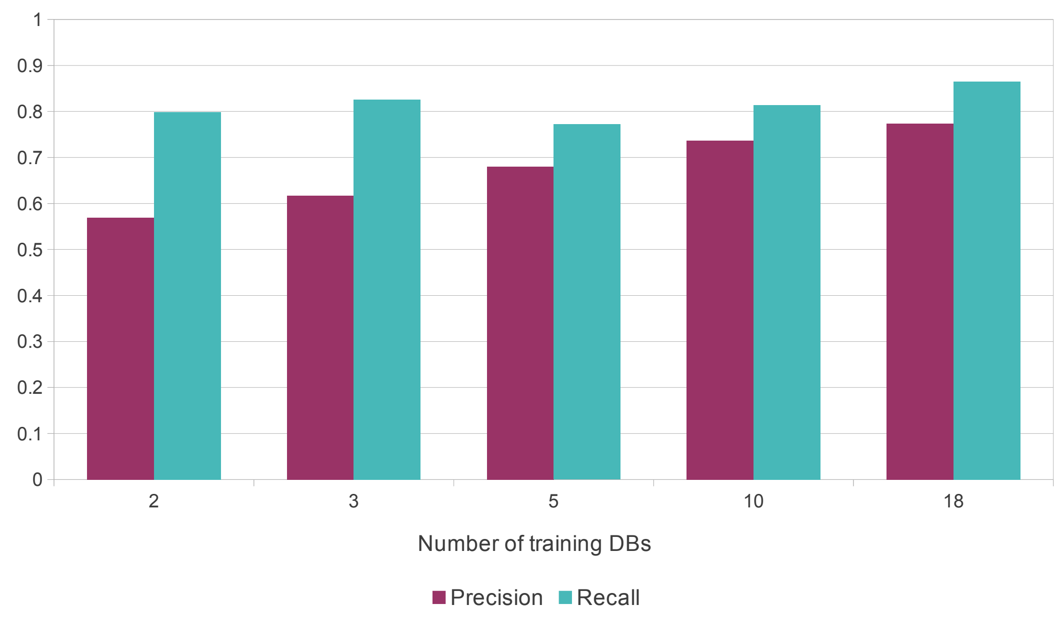


Figure 3.6.: *The precision and recall learning curves for coreference. Almost constant recall when all senses are provided as evidence.*

∅# of formulas	# Queries	# Training DBs	∅F1	∅ Precision	∅ Recall
1651	2	18	0.805	0.781	0.873
1612	10	10	0.722	0.729	0.817
1184	15	5	0.613	0.620	0.748
1008	17	3	0.578	0.558	0.774
788	18	2	0.561	0.533	0.787

(a) Results where symmetric relations are taken into account.

∅# of formulas	# Queries	# Training DBs	∅F1	∅ Precision	∅ Recall
1651	2	18	0.733	0.656	0.888
1612	10	10	0.593	0.514	0.755
1184	15	5	0.450	0.368	0.733
1008	17	3	0.407	0.333	0.719
788	18	2	0.399	0.324	0.738

(b) Results not including symmetric predictions.

Table 3.2.: *Experiments for coreference where no senses are provided as evidence.*

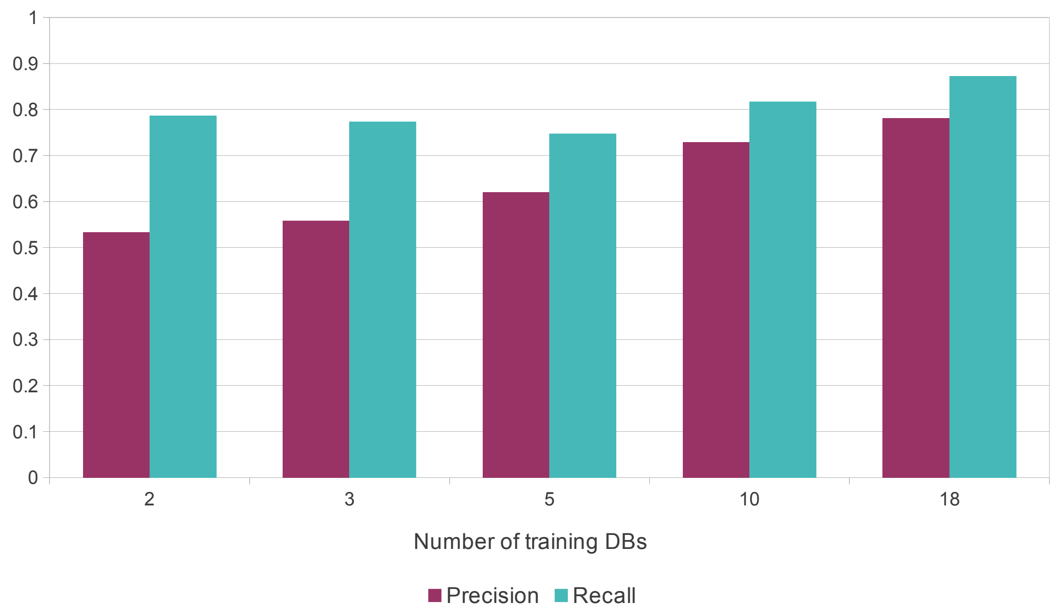
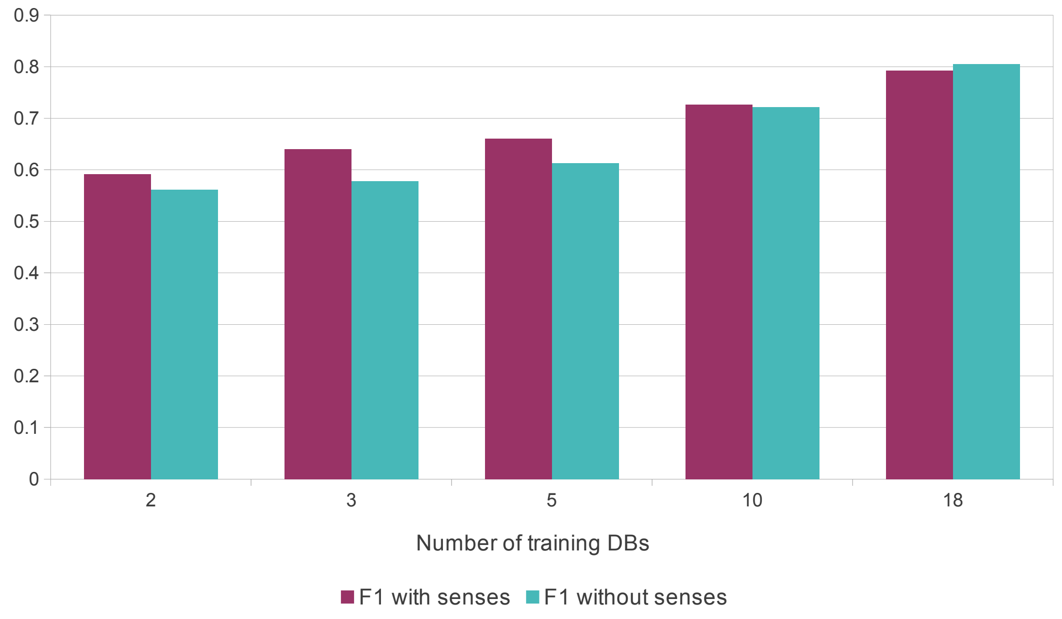


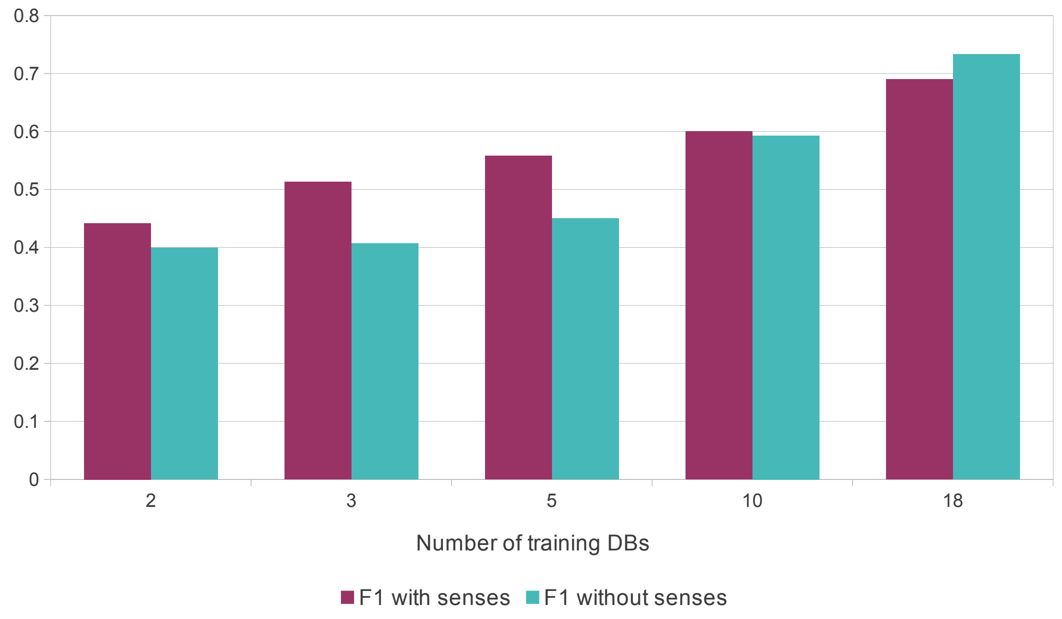
Figure 3.7.: *Precision and recall learning curves for coreference if no senses are given as evidence.*

increase in precision with more training databases is stronger and consequently the difference between highest and lowest precision value is 0.248. Again the reduced results are presented where symmetric relations are filtered out in Table 3.2b. Table 3.2b shows that the model performs a lot worse than in the first experiment when the number of training databases is small.

Comparing the two models, the F1 score is helpful as precision and recall both influence it. Figure 3.8a shows the F1 score for both experiments. As can be seen, not including the senses yields even better results for small training sets when symmetric relations are part of the analysis. Figure 3.8b shows that if the symmetric predictions are left out the model that includes the senses perform a lot better than the one not including senses. However, this is not the case for the set where 18 training databases are provided.



(a) The F1 score for coreference with senses and without senses as evidence.



(b) The F1 scores for coreference with sense and without sense without symmetric predictions.

Figure 3.8.: *Comparing the F1 scores of both experiments.*

0ptGrounding experiments

For the grounding problem two more runs of experiments are conducted. Coreference information is assumed to be provided as evidence and all senses are provided as well. As a metric, only the precision is provided as discussed above.

The same procedure as with the coreference experiments is followed. The set of databases is split into two sets of varying sizes. The results of the experiments are provided in Table 3.3

The first experiment shows satisfying results with a steady increase of precision with an increasing number of training databases.
Despite the fairly good results in the first experiment a second round of experiments is executed where the model is changed by introducing the formula:

isGrounded(w,i) ∧ *hasSense(w,sid)* ∧ *isaW(sid,+c)* ∧ *isInstanceOf(i,cid)* ∧ *isaI(cid,+c)*

The formula is created due to the very low weight that the semantic distance formula receives during training. The effect of this new formula is that for all groundings that appear in the training data the formula will create a large positive weight. If a new object, one that didn't appear during training, is encountered, then the old semantic distance formula still provides it's generalization property. But for objects that haven't been previously seen, the chance of finding correct groundings increases. The results provided in Table 3.4 reflect this intuition in increasing the precision for models that contain a lot of concepts(runs with more than 5 training databases). The learning curves for the two models are provided in Figure 3.9.

∅# of formulas	#Queries	#Training DBs	∅Precision
660	2	18	0.786
661	10	10	0.731
486	15	5	0.724
427	17	3	0.673
319	18	2	0.653

Table 3.3.: *Overview of the grounding experiments. The model shows very good generalization properties for even two training databases show good results for the small training corpus.*

∅# of formulas	#Queries	#Training DBs	∅Precision
843	2	18	0.838
787	10	10	0.801
564	15	5	0.716
505	17	3	0.657
396	18	2	0.605

Table 3.4.: *Better results by making the concepts part of the model*

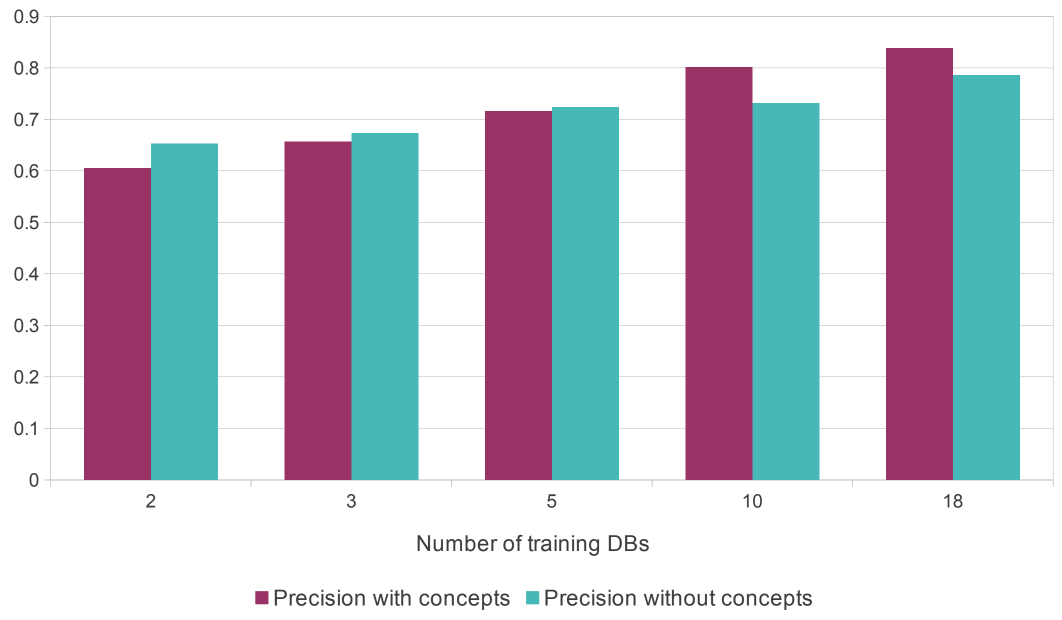


Figure 3.9.: *Precision with concepts in the model versus precision without concepts. The inclusion of concepts only pays off with more training databases.*

0ptJoint experiments

In a third round of experiments the models from coreference and entity resolution are combined into a joint model. The joint model has a much larger complexity since the number of formulas increases dramatically. As a result, not the same amount of data that was obtained. It is indicated how many iterations of the experiments were run in each result table. A single run indicates that 10 models are learned.

Since learning is difficult only one experiment has been conducted for the joint model where '*coreference*' and '*isGrounded*' are query predicates. In the experiment the senses are provided and the concepts are part of the model as in the second grounding experiment. Table 3.5 provides an overview over the obtained data. Table 3.6a and Table 3.7 show the results for the individual queries. Table 3.5b shows the general results if the symmetric predictions for '*coreference*' are removed from the result set.

The data shows that the recall is dramatically reduced compared to the individual experiments. Moreover, there now is a relationship between the number of training databases and the recall, i.e. the recall increases with the number of training databases. Figure 3.10 shows all F1 scores for the '*coreference*' experiments where the word senses are available. As is described above, the first experiments do not include the '*isGrounded*' predicate. Those models tend to show better generalisation properties for less training databases. However, in case of many training databases the joint model seems to outperform the other models.

∅# of formulas	#runs	# Queries	# DBs	∅F1	∅ Precision	∅ Recall
2500	1	2	18	0.800	0.862	0.753
2393	2	10	10	0.610	0.624	0.649
1790	10	15	5	0.566	0.636	0.548
1564	10	17	3	0.561	0.632	0.534
1224	10	18	2	0.545	0.547	0.595

(a) General results for the joint experiment including symmetric predictions.

∅# of formulas	# runs	# Queries	# DBs	∅F1	∅ Precision	∅ Recall
2500	1	2	18	0.826	0.827	0.817
2393	2	10	10	0.561	0.558	0.622
1790	10	15	5	0.542	0.574	0.550
1564	10	17	3	0.516	0.552	0.510
1224	10	18	2	0.443	0.437	0.496

(b) General results without the symmetric predictions from coreference.

Table 3.5.: *General results for the joint experiment.*

# runs	# Queries	# Training DBs	∅F1	∅ Precision	∅ Recall
1	2	18	0.727	0.835	0.655
2	10	10	0.602	0.620	0.678
10	15	5	0.489	0.583	0.504
10	17	3	0.520	0.638	0.519
10	18	2	0.554	0.569	0.690

(a) Results for the coreference predicate with symmetric predictions.

# runs	# Queries	# Training DBs	∅F1	∅ Precision	∅ Recall
1	2	18	0.721	0.784	0.689
2	10	10	0.513	0.478	0.621
10	15	5	0.380	0.447	0.399
10	17	3	0.370	0.451	0.364
10	18	2	0.338	0.354	0.422

(b) Results for the coreference predicate without symmetric predictions.

Table 3.6.: *Results for the coreference predicate.*

# runs	# Queries	# Training DBs	∅ Precision
1	2	18	0.836
2	10	10	0.597
10	15	5	0.611
10	17	3	0.548
10	18	2	0.488

Table 3.7.: *Results for the isGrounded predicate.*

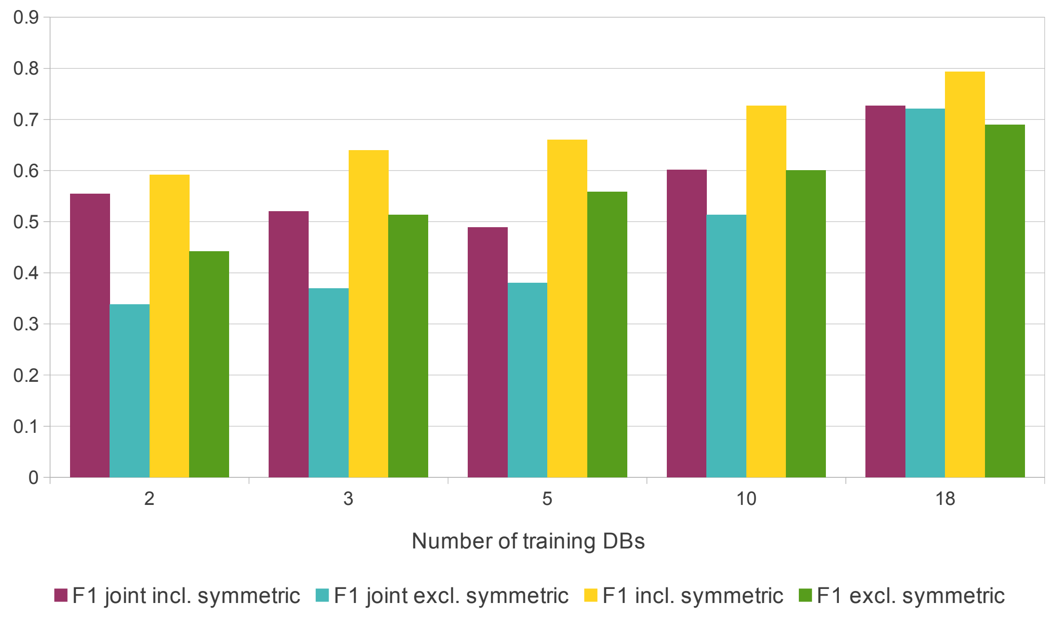


Figure 3.10.: *All F1 scores for the different experiments when word senses are available. Experiments that do not include any grounding information tend to generalize better.*

0ptDiscussion of Results

The results obtained in Section 3.3 generally can be interpreted as a success. The generalisation over the objects in some of the experiments is very promising. Unfortunately, comparison to other models is not possible as models presented in other works are usually tested on different data sets and solve different problems. This is despite the fact that they all deal with language grounding as elaborated on in Section 1.3. However, in this section the strengths and weaknesses of each of the learned models are analysed and the performance compared to each other. Interpretation of the results are provided and ideas for improvement discussed.

0ptDiscussion of coreference results

Coreference resolution has been tested with three different models. The first includes the word senses, the second left the word senses out and then again was tested as part of the joint model where the word senses are present again. Due to time constraints it was not possible to run further tests as they take up very long time(more than a week in case of the joint model).

Despite the fact that comparisons are not directly feasible the author believes that providing the performance metric of at least one reference model is useful as this helps to evaluate the obtained scores. As has been pointed out, state of the art implementations are mostly used with test data from newspaper articles. The implementation described in [**?**] won the CoNLL-2011 shared task competition. Average F1 scores over different test sets vary between 0.57 and 0.74. Hence, the numbers that were obtained in the course of this work seem at least to be competitive. However, it is important to stress that the systems actually test a different tasks and hence cannot be compared directly. The message is that a F1 score of 0.7 can be regarded as a good result.

Some particular interesting results are presented in the last section. First, the almost constant recall that seems to be nearly independent of the number of training databases in the first two experiments. Especially in the case that the symmetric predictions are part of the results. If the symmetric predictions are filtered out then the recall shows greater variance. This is due to the fact that the symmetric predictions are mostly accurate and consequently the number of true positives is reduced when filtering them out. Moreover, the influence of false negatives on the recall increases. The high recall values indicate that '*coreference*' is in general detected when it should be detected. The precision indicates that in case of a lower number of training databases the number of false positives is too high. One way of reducing the number of false positives is introducing more criteria for '*coreference*'. Syntactic relationships that are used in traditional coreference systems could be a start. Nevertheless, this only works for the objects named in the texts and not for the virtual words.

Low precision results for few training databases are most likely obtained due to the fact that not enough possible role combinations are encountered in the data. Many different combinations are required as no generalization mechanism for roles exists in this model which is further discussed in Section 3.5.

Additionally, in the second experiment, where no word senses are provided, the results for many training databases are even better than for the experiment that includes word senses as can be seen in Figure 3.8. This is possibly due to the fact that the semantic distance between two concepts does not provide a secure mechanism to predict '*coreference*' but is just one indicator. In the case that multiple object instances of the same object concept are present the semantic distance does not provide enough information to make a prediction about coreference. This results in an increased number of false positives and hence a lower precision and F1 score.

What is evident from Figure 3.8 is that with five training databases and less, the experiment that excluded word senses shows a weaker results. This in particular the case if symmetric predictions are excluded. This basically shows the powerfulness of the semantic distance relationship which is used for resolving the '*coreference*' predicate.

Although Figure 3.8 indicates a generally lower F1 score for the joint model the model performs fairly well. The low F1 score is mainly due to a much lower recall than in the first experiments. The precision is even higher in the joint approach.

0ptDiscussion of entity resolution results

For the entity resolution problem there are two classes of errors. First, errors that happen are mostly caused by selecting the wrong entity in the right class of objects. For example, in the instruction

'Put the mug on the table.'

In this case the '*mug*' is the '*PutTheme*' of the '*put*' action. The training data suggests that the '*PutTheme*', should never be already on the '*PutGoal*' which is in this case the '*table*'. A wrongly selected item in the right class would be a '*Mug*' that is already located on the table. These kind of errors happen when one sentence requires the object to have certain geometric relationship, e.g. to not be on a table, and all subsequent instructions have a different requirement. This makes sense since the dynamics of the recipe need to be taken into consideration when grounding the objects. However, in the current model, where all sentences are jointly grounded this cannot be accounted for. A possible solution to this approach is discussed in Section 3.5.

The second class of errors happens where misclassification predicts an entirely different class of objects and is lesser a problem, however, still present. This results from the roles that are part of the model. Since no generalisation mechanism exists it is important to have all roles already in the training data so that they can become a part of the model. New roles are not being handled well by the model. Moreover, if the weight for a rule that indicates the grounding of a specific role to a specific concept is very high, then it might overrule the semantic similarity measure.

Making the concepts part of the model improves the precision significantly. This is because for known concepts this boosts the impact of the semantic similarity. The results for the joint experiments are slightly worse than the other ones. It has to be considered that for the first experiment, the coreference relationships were provided as evidence. In the joint approach coreference is part of the inference. Consequently, it is expected that the joint model does not outperform the others. The almost similar precision for 18 training databases suggest that the impact of wrong coreference decisions is low for that many training databases.

0ptDiscussion of the model

The formulas used in the different models are based on the assumptions made in Section 3.2. Consequently, the models can only work within the constraints of the assumptions made. In this section, some of the limitations that are consequences of those assumptions are presented.

The models for coreference as well entity resolution as they are currently implemented and tested in Section 3.3 have several drawbacks that limit the applicability to and scalability to real world scenarios. The most eminent fact is that the current training base only comprises data for five different verbs. This is a big problem although, as mentioned in [**?**], roughly 50% of all actions can be executed with only 15 different verbs. Even if those 15 verbs are explicitly modelled with roles it is hard to find real world examples where only those instructions appear. To overcome this limitation a generalization mechanism is needed that is able to generalize over different verbs. One such approach could be to use a taxonomy for the frames. Such a taxonomy is provided by the FrameNet and could be exploited without having to develop a new taxonomy. However, it is up to new experiments to see if such an approach can still be modelled in ML due to complexity problems when introducing another domain of concepts. In order to handle real-world examples such an approach is necessary and consequently should have a high priority when developing future version of this model.

Moreover, currently the model encodes the action verb that is present in a sentence in the name of a role. This implies that an action verb may only exist once per set of instructions. For example, for the two instructions

Put the cup on the table.
Now put the plate on the table.

The '*table*' in the first and second instruction both would get assigned the role '*Put-Goal*'. Hence, it is not possible to decide to which verb('*put*' in the first or second sentence) the annotated roles belong to. A solution to this problem is to introduce a predicate that takes three parameters. It assigns the roles and also includes the verbs to which the labelled word belongs to.

hasRole(word,word,role!)

Thus, this provides a mechanism to have the role, the action verb and the role assignee in one relation. This has implications for the rest of the model since the complexity of the model will dramatically increase. The modified formula from the coreference model looks like

> $hasRole(w_1, w_2, +r_1) \wedge hasRole(w_3, w_4, +r_2) \wedge coreference(w_1, w_3) \wedge distance0(w_1, w_3)$

Now, four different word variables are needed, instead of the two that were needed previously. For a moderately short example with 16 words and 16 roles this will lead to

$$16^2 \cdot 16^2 \cdot 16^2 = 16,777,216$$

different groundings compared to

$$16 \cdot 16 \cdot 16^2 = 65,536$$

in the current model.

Furthermore, the current model will grow, i.e. the number of formulas, with the number of roles and number of concepts that are present in the training data as pointed out before.

These examples show why it is so hard to create models in ML. Despite the powerfulness of the formalism the exponential growth of the model size is permanently a barrier. Every formula developed in this formalism is generally a trade-off between complexity and expressiveness. This is a big problem when dealing with real world examples where the texts are longer and hence many more words need to be taken into account.

The current model is limited to words that appear in the WordNet taxonomy and therefore dependends on a different project that is outside of the direct influence of the author of this work. However, as has been stated earlier the corpus of objects that is used in everyday manipulation tasks are usually common and can therefore be expected to be part of the taxonomy. To have a more general model for more complex and less mundane tasks the model needs to be changed to be able to handle those unknown objects.

As mentioned in Section 3.1.2 the dynamics of the instructions pose a big challenge for the design of the model. The fact that objects can change their state, can be destroyed or new objects can be created is particularly difficult. In the current model the knowledge about transitions is encoded in the roles of the verb. For example, for a '*mixing sth. with sth.*' action the roles need to include the knowledge that two entities are mixed together and that a resulting entity is created. The problem is that this approach is very inflexible. First, the number of entities that can be mixed together can vary, making it hard to predict the number of parts. Even if the parts can be correctly predicted the next question is if the resulting entity should be in a coreference relationship with its parts. The author argues that for this task a new mereological(part-of) relationship needs to be introduced. Coreference should not be used for this relationship. This is one of the reasons why the problem has been excluded from the current work.

Chapter 4

Conclusion

0ptFuture work

This work lays the basis for further research in the field of language grounding and ML. Many ways to improve the current system and new fields of research have been suggested and are substantiated in this section.

Instruction dynamics. As has been pointed out the transitions that objects undergo in the course of executing a recipe pose a difficult problem. To additionally introduce a new mereological or a more general transformation relationship creates new challenges and the prediction of these relationships in a text is suggested to be tackled in a new project. Moreover, in order to make correct grounding decision in instructions that include these new relationships, it is suggested to process a set of instructions on a per instruction basis. As a consequence, it is suggested to split the joint model of grounding and coreference again into two separate models. This is supported by the data obtained in the experiments as no big improvements could be identified using a joint approach. Moreover, the coreference system should be exploited as evidence for the word sense disambiguation system.

Markov Logic. The algorithms that are available for ML are limited in the applicability in running real world systems. MPE inference is a very fast procedure and gives results even for networks with thousands of variables in only a few seconds. However, learning can take up to several days for big networks and thus hinders the design of bigger models. Each model needs to be tested, refined and relearned several times. Having to wait for several days is not an option for a productive use. Moreover, incremental learning algorithms need to be developed to be able to integrate new knowledge into an existing model without the need to learn over all the historic data. Such algorithms should use already learned weights and incorporate this knowledge into their learning. If a new concept is seen and annotated data is available, the model should be able to integrate a new formula into the existing model without having to recompute the entire model. With this method, constant learning could be achieved and make the use in productive environments more realistic. Even if learning takes up several hours, the robot could do this overnight.

Training data. The lack to annotated training data is a big problem. Despite the fact that the models generalize fairly well, lots of training data is needed to stabilize the system in real world scenarios. Although an annotation tool has been created that makes annotating texts and exporting them to the ML format easy, it takes some training for the annotator to consistently annotate the data. Especially annotating the roles requires a deep understanding of the semantics of the roles and experiments need to be conducted in order to see if crowd sourcing, e.g. with Amazon's mechanical Turk is feasible.

0ptSummary

This work presents a novel approach to language grounding using background knowledge. Section 1.2 approaches this new concept introducing virtual words and how action specific models can be used to identify the needed items necessary for a successful task execution on an autonomous robot. Chapter 2 equips the reader with the necessary background information to understand the models that are developed. Specifically, the PRAC formalism is introduced in Section 2.1 that is used as the basis of this work. ML is used as a probabilistic first order knowledge base that can be queried in order to infer missing information. Models for coreference as well as for entity resolution are developed in Section 3.1. A joint model combines the two models into a single model. In order to run several experiments a set of training databases is created. Several experiments with the different models are conducted and the results presented in Section 3.3. The results prove the general applicability of the chosen approach in a limited setting with a reduced complexity of the environment as well as short instructions. A critical analysis of the results is given in Section 3.4. The model for coreference achieves in the best case an F1 score of 0.793. The model for entity

resolution achieves an F1 score 0.805 in the best case on the provided test set. The joint model achieves an F1 score of 0.80 for the combined task. Moreover, general model assumptions are analysed in Section 3.5 and implications of modelling decisions ascertained. It is further shown in Section 4.1 how this work is the basis for future research in the area of language grounding for autonomous robots. The natural handling of instruction dynamics is among the most relevant issues for future work.

In conclusion, this work provides new ideas for language grounding and additionally, it provides a new approach to classical NLP problems. Due to the positive results of the experiments conducted in the scope of this work, the author expects more research being conducted using semantic knowledge for instruction processing and NLP tasks in general.

Appendix A

Training Database

The CD-ROM submitted with this work includes all experimental data. See “readme.txt” for an explanation of the file structure.

OptTexts

Text 1.

1: First mix orange juice with vodka in the shaker.
2: Add a straw.
3: Serve.

Text 1.

1: Add Whiskey and ginger ale to a highball glass.
2: Start mixing.
3: Serve cold.

Text 3.

1: First mix the pepper with the salad.
2: Serve.

Text 4.

1: Add oatmeal and cinnamon to a bowl.
2: Now mix.
3: Serve.

Text 5.

1: Add coca cola to a glass.
2: Afterwards mix with ice.

3: Serve on a tray.

Text 6.

1: Put the fork on the napkin.

Text 7.

1: Put the saucepan on the stove.
2: Add spaghetti and salt.
3: Serve on a plate.

Text 8.

1: Add cheese and toast on a platter.
2: Now put the platter on the table.

Text 9.

1: Put rice in a steamer.
2: Add water.
3: Mix.
4: Serve.

Text 10.

1: Put the cup on the table.
2: Add coffee to the cup.
3: Mix with milk.
4: Serve.

Text 11.

1: Put couscous into a pot.
2: Add double the volume of water.
3: Serve when finished.

Text 12.

1: First mix strawberries with cereal with yoghurt.
2: Add milk.
3: Serve.

Text 13.

1: Put a vase on the table.
2: Add flowers to the vase.

3: Then mix with fertilizer.
4: Serve.

Text 14.

1: Fill a coffee mug with water.
2: Add instant-coffee.
3: Serve.

Text 15.

1: Fill beer in a jug.
2: Put a beer glass on the bar.
3: Add beer to the beer-glass.

Text 16.

1: Fill a bottle with soda-water.
2: Add lime.
3: Serve.

Text 17.

1: Put smoked-salmon and bread on a buffet.
2: Fill a jar with cookies.

Text 18.

1: First put a mug and a spoon on the kitchen table.
2: Fill the mug with apple-juice.

Text 19.

1: Fill dark chocolate in a chocolate-fountain.
2: Add cherries.
3: Serve.

Text 20.

1: Mix apple, bananas and oranges in a bowl.
2: Add sugar.
3: Serve in a bowl.

0ptAction verb models

to mix.

- MixPart
  Identifies one of the parts mixed
- MixWhole
  The resulting entity of the mixing operation
- MixPlace
  The location where the mixing occurs

to add.

- AddPlace
  The place where the *AddNewMember* is added into an existing group
- AddNewNewMember
  An item that becomes part of the AddGroup
- AddGroup
  Anything that can be conceptualized as a complex collection of parts or ingredients. Can also be empty.
- AddExistingMember
  An existing member of the *AddGroup*

to serve.

- ServeTheme
  The object that is served
- ServeMeans
  The means by which the *ServeTheme* is served

to put.

- PutTheme
  The object that is moved
- PutGoal
  The location where the *PutTheme* is relocated to

to fill.

- FillTheme
  The entity that is filled
- FillGoal
  The location the *FillTheme* is filled into

Appendix B

Markov Logic Models

The ML models are provided here. Predicate definitions are left out.

0ptCoreference resolution

```
coreference(word,word)
hasSense(word,sense!)
hasRole(word,role!)
isaW(sense, concept)
distance0(word,word)
distance1(word,word)
distance2(word,word)
distance3(word,word)
fardistance(word,word)

0 hasSense(w1,sid1) ∧ hasSense(w2,sid2) ∧ isaW(sid1,c) ∧ isaW(sid2,c) ∧
coreference(w1,w2)
0 hasRole(w1, +r1) ∧ hasRole(w2, +r2) ∧ coreference(w1,w2) ∧distance0(w1,w2)
0 hasRole(w1, +r1) ∧ hasRole(w2, +r2) ∧ coreference(w1,w2) ∧distance2(w1,w2)
0 hasRole(w1, +r1) ∧ hasRole(w2, +r2) ∧ coreference(w1,w2) ∧distance3(w1,w2)
0 hasRole(w1, +r1) ∧ hasRole(w2, +r2) ∧ coreference(w1,w2) ∧fardistance(w1,w2)
0 hasRole(w1, +r1) ∧ hasRole(w2, +r2) ∧ coreference(w1,w2) ∧distance1(w1,w2)
coreference(w1,w3) => (coreference(w1,w2) <=> coreference(w2,w3)).
0 coreference(w1,w2) ∧ hasPOS(w1,+pos1) ∧ hasPOS(w2, +pos2)
0 hasSense(w1,NULL) => coreference(w1 ,w2)
```

0ptEntity resolution

```
isGrounded(word,instance!)
hasSense(word,sense!)
hasRole(word, role!)
coreference(word,word)
isaW(sense,concept)
isaI(cid,concept)
isInstanceOf(instance, cid)
on(instance,instance)
in(instance,instance)
distance0(word,word)
distance1(word,word)
distance2(word,word)
distance3(word,word)
fardistance(word,word)

0 isGrounded(w,i) ∧ hasSense(w,sid) ∧ isaW(sid, c) ∧ isInstanceOf(i,cid) ∧
isaI(cid, c)
0 isGrounded(w,i) ∧ hasSense(w,sid) ∧ isaW(sid, +c) ∧ isInstanceOf(i,cid) ∧
isaI(cid, +c)
0 isGrounded(w1,i1) ∧ isGrounded(w2,i2) ∧ hasRole(w1,+r1) ∧ hasRole(w2,+r2)
∧ on(i1,i2)
0 isGrounded(w1,i1) ∧ isGrounded(w2,i2) ∧ hasRole(w1,+r1) ∧ hasRole(w2,+r2)
∧ in(i1,i2)
coreference(w1,w2) => (isGrounded(w1,i) <=> isGrounded(w2,i)).
hasSense(w,NULL) => isGrounded(w,NULL).
0 hasPOS(w, +pos) => isGrounded(w,NULL)
0 !(hasPOS(w,+pos) => isGrounded(w,NULL))
0 hasRole(w,+r) => isGrounded(w,NULL)
```

0ptJoint model

```
isGrounded(word,instance!)
hasSense(word,sense!)
```

```
 3: hasRole(word, role!)
 4: coreference(word,word)
 5: isaW(sense,concept)
 6: isaI(cid,concept)
 7: isInstanceOf(instance, cid)
 8: on(instance,instance)
 9: in(instance,instance)
10: distance0(word,word)
11: distance1(word,word)
12: distance2(word,word)
13: distance3(word,word)
14: fardistance(word,word)
15:
16: 0 hasSense(w1,sid1) ∧ hasSense(w2,sid2) ∧ isaW(sid1,c) ∧ isaW(sid2,c) ∧
    coreference(w1,w2)
17: 0 isGrounded(w,i) ∧ hasSense(w,sid) ∧ isaW(sid, c) ∧ isInstanceOf(i,cid) ∧
    isaI(cid, c)
18: 0 isGrounded(w,i) ∧ hasSense(w,sid) ∧ isaW(sid, +c) ∧ isInstanceOf(i,cid) ∧
    isaI(cid, +c)
19: 0 hasRole(w1, +r1) ∧ hasRole(w2, +r2) ∧ coreference(w1,w2) ∧ distance0(w1,w2)

20: 0 hasRole(w1, +r1) ∧ hasRole(w2, +r2) ∧ coreference(w1,w2) ∧ distance2(w1,w2)

21: 0 hasRole(w1, +r1) ∧ hasRole(w2, +r2) ∧ coreference(w1,w2) ∧ distance3(w1,w2)

22: 0 hasRole(w1, +r1) ∧ hasRole(w2, +r2) ∧ coreference(w1,w2) ∧ fardistance(w1,w2)

23: 0 hasRole(w1, +r1) ∧ hasRole(w2, +r2) ∧ coreference(w1,w2) ∧ distance1(w1,w2)

24: 0 coreference(w1,w2) ∧ hasPOS(w1,+pos1) ∧ hasPOS(w2, +pos2)
25: 0 hasSense(w1,NULL) => coreference(w1 ,w2)
26: 0 isGrounded(w1,i1) ∧ isGrounded(w2,i2) ∧ hasRole(w1,+r1) ∧ hasRole(w2,+r2)
    ∧ on(i1,i2)
```

```
27: 0 isGrounded(w1,i1) ∧ isGrounded(w2,i2) ∧ hasRole(w1,+r1) ∧ hasRole(w2,+r2)
    ∧ in(i1,i2)
28: coreference(w1,w3) => (coreference(w1,w2) <=> coreference(w2,w3)).
29: coreference(w1,w2) => (isGrounded(w1,i) <=> isGrounded(w2,i)).
30: hasSense(w,NULL) => isGrounded(w,NULL).
31: 0 hasPOS(w, +pos) => isGrounded(w,NULL)
32: 0 !(hasPOS(w,+pos) => isGrounded(w,NULL))
33: 0 hasRole(w,+r) => isGrounded(w,NULL)
```