Universidade de Aveiro Departamento de Electrónica, Telecomunicações e
2014 Informática

Aneesh Chauhan

Formação do Significado Perceptual das Palavras através de Interacção

Grounding Human Vocabulary in Robot Perception through Interaction

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2014

Aneesh Chauhan

Formação do Significado Perceptual das Palavras através de Interacção

Grounding Human Vocabulary in Robot Perception through Interaction

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Informática, realizada sob a orientação científica do Doutor Luís Seabra Lopes, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

This work was supported by the PhD scholarship grant awarded by the Portuguese Foundation for Science and Technology (FCT) under contract: SFRH/BD/31577/2006

I dedicate this work to my parents Sh. Balak Ram Chauhan and Smt. Kala Chauhan. All I have accomplished was possible because of their love, sacrifices and unflinching support.

o júri

presidente

Doutor Fernando Joaquim Fernandes Tavares Rocha
Professor Catedrático do Departamento de Geociências da Universidade do Aveiro

Doutor Pascual Campoy Cervera
Professor Catedrático do Departamento de Automática, Ingeniería Electrónica e Informática Industrial da Universidad Politécnica de Madrid

Doutor José Santos-Victor
Professor Catedrático do Instituto Superior Técnico da Universidade de Lisboa

Doutor João Manuel Portela da Gama
Professor Associado da Faculdade de Economia da Universidade do Porto

Doutora Ana Maria Perfeito Tomé
Professora Associada do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Doutor Luís Filipe de Seabra Lopes
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro (Orientador)

agradecimentos

Firstly I would like to thank my thesis supervisor Prof. Luís Seabra Lopes. He has played a huge part in my development as a researcher, and his thoughts, advises, and constructive criticism will continue to guide me beyond this thesis. His efforts in shaping my PhD, including this document, were absolutely invaluable.

I want to thank the Portuguese Science Foundation (FCT) for the PhD scholarship grant. I am very thankful to IEETA, my research institution, for providing the financial support for attending several conference events, and IAPR society for monetary help for attending ICPR2012 in Tsukuba, Japan. I would also like to thank the open-source community for sharing and spreading the knowledge, for all to use.

Two colleagues have contributed to the part of the work in this thesis. Amanda Nascimento contributed to the development of the component-based representation, an extended version of which became part of this thesis. Bruno Werneck contributed to the development of software for the robotic-arm control.

I want to thank little boy José Diogo, who contributed to science by letting me use his toys (which I used for teaching the robotic agent, and for building the LANGG68 dataset).

I am extremely thankful to Bruno Pimentel, Marcelo Quinderé and Mário Rodrigues for their friendship, guidance, stimulating conversations and help when I needed. Bruno, your support in good and tough times has been incredible. I am lucky to have you as a friend. I was fortunate to have Marcelo as my colleague in 2005, in a new country and a new lab. His technical knowledge and his patience with my questions have helped me greatly. I have nagged Mario, for years, with mathematics, technology and everything in between. I have enjoyed my discussions with you and learned a lot from you.

I want to thank Conni, Eduardo, Jenni, Alina, Mario Antunes, Rui Perreira, Luís Ribeiro, Eurico and João Cunha for their friendship and for keeping me sane over the years. You guys have been amazing and I am so glad to have come across you. Miguel Oliveira, I am very thankful to you for helping me with the experiments, after I had left Aveiro. In my absence from Aveiro, Alina and Jenni ended up taking care of a lot of paperwork, for which I am very thankful.

Elena, your patience, strength and uncommon ability to empathize has made this journey possible for me. You have suffered with me and for me. I find myself very fortunate to have you by my side. Thank you so much.

Finally, I would like to thank my parents, my grandmother and my sister for their constant encouragement, strength and support. This thesis, in many ways, is a result of the dedication of my parents to the education of their children. Mere words cannot suffice to express my gratitude.

palavras-chave

aquisição de vocabulário, aprendizagem de categorias, arquitecturas de aprendizagem, interação homem-robô, percepção visual, metacognição.

resumo

Esta tese aborda o problema da aprendizagem de palavras em agentes computacionais. A motivação por trás deste trabalho reside na necessidade de suportar a comunicação baseada em linguagem entre os robôs de serviço e os seus utilizadores humanos, bem como suportar o raciocínio baseado em símbolos que sejam relevantes no contexto das tarefas atribuídas e cujo significado seja definido com base na experiência perceptiva. Mais especificamente, o foco da investigação é o problema de estabelecer o significado das palavras na percepção do robô através da interacção homem-robô.

A definição do significado das palavras com base em experiências perceptuais e perceptuo-motoras enfatiza o papel da configuração física e perceptuo-motora do robô. Entretanto, a língua é um produto cultural criado e adquirido através de interacções sociais. Isso destaca o papel da sociedade como fonte linguística. Tendo em conta estes aspectos, um cenário experimental foi definido no qual um instrutor humano ensina a um agente robótico os nomes dos objectos presentes num ambiente visualmente partilhado. O agente associa os nomes desses objectos à sua percepção visual desses objectos.

A aprendizagem de palavras é um problema sem objectivo pré-estabelecido. Nós adquirimos novas palavras ao longo das nossas vidas. Assim, a arquitectura de aprendizagem do agente deve poder adquirir palavras e categorias de uma forma semelhante. Neste trabalho foram concebidas quatro arquitecturas de aprendizagem que podem ser usadas por agentes robóticos para aprendizagem e aquisição de novas palavras e categorias, incrementalmente. Os métodos de aprendizagem utilizados nestas arquitecturas foram projectados para funcionar de forma incremental, acumulando um conjunto cada vez maior de palavras e categorias.

É proposta e aplicada uma nova metodologia da avaliação experimental que leva em conta a natureza aberta e incremental da aprendizagem de palavras. Esta metodologia leva em consideração a constatação de que o vocabulário de um robô será limitado pela sua capacidade de discriminação, a qual, por sua vez, depende dos seus sensores e capacidades perceptuais. Foi realizado um extenso conjunto de experiências sistemáticas em múltiplas situações experimentais, para avaliar cuidadosamente estas abordagens de aprendizagem. Os resultados indicam que todas as abordagens foram capazes de adquirir novas palavras e categorias incrementalmente. Embora em algumas das abordagens não tenha sido possível atingir vocabulários maiores, verificou-se que uma das abordagens conseguiu aprender até 293 categorias, com potencial para aprender muitas mais.

keywords

vocabulary acquisition, open-ended category learning, learning architectures, language grounding, human-robot interaction, visual perception, metacognition

abstract

This thesis addresses the problem of word learning in computational agents. The motivation behind this work lies in the need to support language-based communication between service robots and their human users, as well as grounded reasoning using symbols relevant for the assigned tasks. The research focuses on the problem of grounding human vocabulary in robotic agent's sensori-motor perception.

Words have to be grounded in bodily experiences, which emphasizes the role of appropriate embodiments. On the other hand, language is a cultural product created and acquired through social interactions. This emphasizes the role of society as a source of linguistic input. Taking these aspects into account, an experimental scenario is set up where a human instructor teaches a robotic agent the names of the objects present in a visually shared environment. The agent grounds the names of these objects in visual perception.

Word learning is an open-ended problem. Therefore, the learning architecture of the agent will have to be able to acquire words and categories in an open-ended manner. In this work, four learning architectures were designed that can be used by robotic agents for long-term and open-ended word and category acquisition. The learning methods used in these architectures are designed for incrementally scaling-up to larger sets of words and categories.

A novel experimental evaluation methodology, that takes into account the open-ended nature of word learning, is proposed and applied. This methodology is based on the realization that a robot's vocabulary will be limited by its discriminatory capacity which, in turn, depends on its sensors and perceptual capabilities. An extensive set of systematic experiments, in multiple experimental settings, was carried out to thoroughly evaluate the described learning approaches. The results indicate that all approaches were able to incrementally acquire new words and categories. Although some of the approaches could not scale-up to larger vocabularies, one approach was shown to learn up to 293 categories, with potential for learning many more.

Contents

List of Figures

List of Tables

List of Abbreviations

AVG	Average classifier rule
DTW	Dynamic Time Warping
ES	Euclidean Similarity
GSW	Grounding Spoken Words architecture
HRI	Human-Robot Interaction
IBL	Instance-based learning
MCML	Multiple classifiers with meta-learning
MCS	Maximum Common Subgraph
MD	Manhattan Distance
MFC	Mel-Frequency Cepstrum
MFCC	Mel-Frequency Cepstral Coefficients
MPD	Manhattan Pyramid Distance
MPS	Manhattan Pyramid Similarity
NDC	Normalized Distance to the Center
NC	Nearest Cluster classifier rule
NN	Nearest Neighbor classifier rule
$\tilde{P}_{\Delta}$	Un-normalized Pyramid Match
P_{Δ}	Normalized Pyramid Match
PMK	Pyramid Match Kernel
RADSD	Normalized Radius Standard Deviation

SLH	Shape Layers Histogram
SSNRA	Shape Slices Normalized Radii Averages
SSNRSD	Shape Slices Normalized Radii Standard Deviations
SSH	Shape Slices Histogram
SVDD	Support Vector Data Description

1 Introduction

Robots are expected to adapt to the non-expert user. This adaptation includes the capacity to take a high-level description of the assigned task and carry out the necessary reasoning steps to determine exactly what must be done. Adapting to the user also implies using the communication modalities of the user. Spoken language is probably the most powerful communication modality. It can reduce the problem of assigning a task to the robot to a simple sentence, and it can also play a major role in teaching the robot new facts and behaviors. There is, therefore, a trend to develop robots with spoken language capabilities for human-robot interaction (see Baxter et al., 2011; Breazeal, 2003; Fong et al., 2003; Goodrich and Schultz, 2007; Hegel et al., 2007; Murphy et al., 2010; Seabra Lopes, 2002; Seabra Lopes and Chauhan, 2008; Seabra Lopes and Connell, 2001; Spexard et al., 2007; Thomaz and Breazeal, 2006, and many others).

In recent years, there has been a significant progress towards designing conversational agents capable of holding a dialog with one or more human participants in relatively unrestricted environments (Bohus and Horwitz, 2009; Gold et al., 2009; Mutlu et al., 2009). Apart from focusing on various communication challenges (e.g. speech recognition, voice synthesis), most of these approaches take cues from human-to-human discourse behavior (eye gaze movements, hand gestures, selective attention, target recognition and tracking etc.) to build robotic agents that can converse in a human-like manner.

Although such systems have been shown to be robust in a variety of real world scenarios, they lack the semantic perception of their language of communication. That is, these systems operate on the language symbols (words) and produce a reply. The meaning interpreted from these replies lies inside the head of the person interpreting them and the algorithm designer, but not inside the computer manipulating these symbols (Harnad, 1990; Searle, 1980). To give conversational robots increased cognitive plausibility, it is essential that they have the capacity to ground human language in their perception. This thesis is a product of efforts in this direction.

1.1 Situating the problem

Language processing, like reasoning capabilities, involves the manipulation of symbols. By symbol it is meant a pattern that represents some entity in the world by association, resemblance or convention (Seabra Lopes and Chauhan, 2007). Association and resemblance arise from perceptual, sensori-motor and functional aspects, whereas convention is socially or culturally established.

The advent of computers encouraged people to start developing "intelligent" artifacts, including artifacts with human-level intelligence (Turing, 1950). As reasoning and language are key components of intelligence, the first few decades of research on artificial intelligence (AI) focused on first-order logic, semantic networks, logical inference, search techniques and natural language processing. Symbol systems in AI were theorized by Simon and Newell (Newell and Simon, 1972; Simon, 1979) in successive publications since the 1970s, and became the dominant model of the mind in cognitive science (see the survey and critical analysis of Anderson and Perlis (2002)). In classical artificial intelligence, symbolic representations were amodal in the sense that they had no obvious correspondence or resemblance to their referents (Barsalou, 1999).

In itself, a symbol does not contain a meaning. Its meaning lies in its association with the entity of the world it refers to (Barsalou, 1999; Harnad, 1990). Since the role of perception and sensori-motor control, in classic symbolic AI, was largely overlooked, connecting symbols and to their real-world referents remained an open issue. The problem of associating a word (or a symbol) to its referent is known as the symbol grounding problem (Harnad, 1990). Over the years, many researchers have increasingly argued for the role of the physical body in language acquisition, where, experiences from sensory-motor perception and control of the environment are considered a necessary part of semantic representations. This in literature is generally known as situated or embodied AI.

The increasing concern with perception and sensorimotor control, both in the AI and robotics communities, was paralleled in cognitive science. Barsalou (1999) developed a theory on "perceptual symbol systems", which takes up the classical (perceptual) view of cognition. A 'perceptual symbol' is viewed as an unconscious neural representation that represents some component of perceptual experience. Related perceptual symbols become organized into a kind of category or concept, called a simulator. The simulator is able to produce limitless simulations (conscious mental images of members of the category) even in the absence of specific perceptual experience. Simulators can be aggregated in frames to produce simulators for more complex categories. Linguistic symbols are viewed as perceptual symbols for spoken or written words. As linguistic simulators develop, they become associated with the simulators of the entities to which they refer. Although the "simulation" account of grounding concrete words (and embodied cognition) has gained strong support over the years, this approach is considered problematic for grounding abstract vocabulary and abstract concepts (Barsalou, 1999). More recently, new theories based on mental simulations have argued for the important role of affective processes (in addition to other sensory-motor experiences) in grounding abstract vocabulary (Vigliocco et al., 2009).

Taking a broader perspective, Clark (1997) saw control of embodied action as an emergent property of a distributed system composed of brain, body and environment. However, Clark rejected radical anti-representationalist approaches and argued for the need of representations geared to specific sensory-motor needs. He also emphasized the importance of external scaffolding, that is, the support provided to thought by the environment and by public language.

Analogies with formal symbol systems and computer languages have led many to treat human language as a code, that is, a determinate set of tokens manipulated according to a determinate set of rules. By contrast, a distributed view on language origins, evolution and acquisition is emerging in linguistics. In this new perspective, language is treated as a cultural product, perpetually open-ended, incomplete and ambiguous to some extent (Love, 2004). Rather than being an internal code, language is an external cognitive tool that simultaneously reflects cultural conceptualizations of the world and helps to create internal conceptualizations in individuals. The study of language origins and evolution has been performed using multi-robot models, with the Talking Heads experiments as a notable example (Steels, 2001, 2003). In this case language is transmitted horizontally in the population of robots. Meanwhile, processes where language is vertically transmitted are of particular relevance to robotics applications. In vertical transmission, an agent or population of agents inherits most of its linguistic behavior from a previous generation, or from an independent population (Kirby and Hurford, 2002; Steels and Kaplan, 2002). Given that language acquisition and evolution, both in human and artificial agents, involve not only internal, but also cultural, social and affective processes, the underlying mechanism has been called "external symbol grounding" (Belpaeme and Cowley, 2007; Cowley, 2006). These studies are reaching towards a common ground with respect to the conclusions reached in cognitive science - 'language and cognitive dynamics are mutually constitutive' (Belpaeme and Cowley, 2007).

These studies have helped to bring forward many new insights into language emergence and acquisition in humans. In particular, as can be observed from the discussion above, in the debate on language grounding, three main factors play central role:

– **Meaning formation** : Set of cognitive capabilities that allow an individual to represent entities of the world in its brain.

– **Embodiments** : Linguistic symbols should be grounded in sensori-motor perception.

– **Society** : Society is the source of these linguistic symbols and social interactions are at the core of language transfer;

These insights have been equally valuable for designing computational agents (robotic or otherwise) for grounding language in perception. These agents are set in social settings, where humans or other agents are the source of linguistic input. Agents are designed with specific embodiments which allow them to sense the environment as well as act upon it. Different learning and classification methods are implemented over these agents, which allow them to learn sensori-motor categories and associated vocabulary. That is, the factors outlined above are explicitly considered in designing such agents. However, with respect to the implementation details, there is a huge variety. The social setting, the types of embodiments and the learning approaches differ from one approach to another. Different computational models for language grounding and their underlying specificities will be reviewed in the next chapter in Section 2.3.

The global aim of the research conducted in this thesis is to develop one such agent which can acquire the human vocabulary by interacting with the human users in a shared social setting. This agent should be supported by appropriate physical and cognitive capabilities (*i.e.* suitable sensori-motor embodiments and learning and classification methods) that allow it to ground these words in its sensori-motor perception.

1.2 Thesis scope and approach: taking baby steps

Most of the work in last decades has been on designing agents that are able to ground vocabulary (object names) in visual perception (see the survey in Section 2.3). Although, more recently, some works have begun to investigate grounding of rudimentary syntax as well (Chella et al., 2009; Dindo and Zambuto, 2009). In this thesis, as in most other works reported in the literature, vocabulary acquisition will be explored in visual domain. Such popular choice is justified by analogies with early language development in children.

Words are at the core of the language understanding and acquisition processes. Before the infants are able to grasp the understanding of rules associated with their native languages, they are already accumulating lexicon at a gradually increasing rate. The early lexicon in children (till the age of 3) consists mainly of common nouns that name concrete objects in the children's environment (e.g. toys, food items, geometric shapes, animal categories) and to a lesser extent routine social words, proper nouns, animal sounds and observable verbs (Bloom, 2001; Messer, 1994). The overwhelming bias towards learning names of concrete object categories is a direct consequence of the early conceptual development process in infants. Studies have shown that infants at a very young age start showing an attentional bias towards categories that have clearly defined shapes (Bomba and Siqueland, 1983; Landau et al., 1988; Smith and Samuelson, 2006). This makes grounding the names of visually concrete referents an easier task (Gillette et al., 1999) and their early existence prevalent in comparison to other words. In later years of development, it has been suggested in literature, this basic vocabulary is recruited by infants to incrementally acquire more complex aspects of their native language (Gleitman and Landau, 1994; Pinker, 1984).

The significance of shape information in early word learning led us to explore multiple approaches to represent the shape of an object. In the context of this thesis, multiple novel methods were developed for describing the shape of an object (described in Chapter 4). By deriving the shape information of an object from its contour multiple representations were designed (these representations are akin to the shape signatures of Zhang and Lu (2004)). Additionally, by finding the primitive components of an object using only color information, a graph-based shape representation was also developed where the graph is derived from the spatial arrangement of these primitive components.

Word learning is an open-ended problem. Our cognitive capabilities allow us to acquire new words and new categories throughout our lives. Taking an example from early language development, vocabulary in children starts with about 10 words in the first year and reaches to almost 300 words by the end of the second year (Bates et al., 1992; Bloom, 2000; Crystal, 1987; Fenson et al., 1994). This means, infants (and humans in general) are endowed with innate cognitive capabilities to allow incremental acquisition of vocabulary and the associated visual concepts.

Language grounding is highly dependent on the techniques and methods being used for learning. While keeping in mind the distributed/extended/social nature of language acquisition, this thesis focuses on the required internal inference and memory processes for open-ended category learning and vocabulary acquisition. The emphasis on open-endedness is justified by two main factors. On one hand, human language is open-ended, as mentioned. So, it is not viable to predefine a vocabulary and a corresponding set of categories. On the other hand, robots currently are rather limited in their perceptual and sensori-motor abilities, which prevents them from acquiring large subsets of a human language. In this context, open-ended category learning is necessary to support the adaptation of robots with limited abilities to specific users, tasks and environments. However, very few researchers have investigated agent architectures supporting open-ended category learning, symbol grounding and language acquisition (see Kirstein and Wersing, 2011; Kirstein et al., 2012; Skočaj et al., 2007, for exceptions).

In this thesis, four different open-ended learning architectures were explored, where the key difference between these architectures is the learning and classification approach used:

1. Architecture based on instance-based learning with a single classifier (Section 3.3.1): In this architecture, the category learning approach is instance-based and categories are simply described by the instances belonging to that category (Section 5.1). Additionally, a cluster-based approach to represent categories is also explored, which is a modification of the simple instance-based approach (Section 5.2). The cluster-based approach is still instance-based, since all individual instances are still stored in memory and used for classifiying new objects. Classification in both cases is based on defining a measure of membership between an object to be classified and a category description. Sections 5.1.2 and 5.2.2 present the set of category membership measures.

2. Architecture based on one-class learning (Section 3.3.2): In this architecture, the learning paradigm of choice is one-class learning. The motivation behind this preference is to imitate the language development process in children at the single-word stage. Studies in cognitive language development literature indicate that children predominantly learn from positive examples (Bloom, 2000; Markman, 1989). One-class classifiers, which learn only from positive examples, seem especially suited for such problems. Therefore one of the learning approaches investigated in this thesis is based on support vector data descriptions, SVDD (Tax, 2001). SVDD is a single-class classifier that has been shown to be robust at novelty detection tasks using only a few positive examples Therefore, an incremental learning system based on SVDD classifiers was developed to support open-ended category learning and vocabulary acquisition. Converting original implementation of SVDD to suit online, incremental and open-ended problems led to several novelties, as described in Section 5.3.3.

3. Architecture based on multiple classifiers and meta-learning (Section 3.3.3): In this architecture, an instance-based approach, with simple feature spaces, is also adopted for category representation. Adequate classification relies on two main ingredients: similarity assessment based on multi-resolution matching; and a metacognitive self-monitoring and control loop. Multiple object representations and multiple classifiers and classifier combinations are used. All learning computations are carried out during the normal execution of the agent, which allows continuous monitoring of the performance of the different classifiers. The measured classification successes of the base classifiers are used to reconfigure dynamically some of the classifier combinations as well as to select the classifier that will be used to predict the category of a new unseen object. This approach is described in Section 5.4.

4. Architecture for grounding spoken words (Section 3.4): The first three architectures were designed to ground textual words. In this architecture, words are communicated via speech. Unlike textual input, the speech signal is ambiguous. This poses additional problems and requires an integrated learning architecture that collates learning

of spoken word categories with learning of visual categories. An unsupervised clustering method is used for learning spoken word categories. Learning word categories leads to dynamic formation and reorganization of object categories. Classification of object instances is supported by the approach used in the architecture based on multiple classifiers systems. The approach is detailed in Chapter 6.

"Meaning formation", as discussed in the previous section, is one of the key factors that needs to be addressed for designing agents that ground words. The learning approaches presented in this thesis specifically address this issue. These architectures and the corresponding learning approaches are designed for long-term and open-ended learning of visual categories and category names.

Another main factor to be addressed is "embodiment". To support the experimental work, a simple agent was developed. It consists of appropriate sensori-motor components that help it interact with and perceive its environment. Since the focus of the thesis was on visual perception, the main sensing device is a camera. The agent is also able to act upon its environment using a robotic arm. Additionally, for the architecture supporting spoken words, a microphone is used for supporting speech input. An attached computer runs appropriate perceptual, learning and interaction procedures. Thorough details of the agent's complete architecture are presented in Chapter 3.

The final key factor to be accounted for is "social interaction". A simple social language grounding experimental setup is designed where a human instructor teaches a robotic agent the names of the objects present in a visually shared environment. The agent's world includes a user, a visually observable area and real-world objects whose names the user may wish to teach. The user, who is typically not visible to the agent, acts as the language instructor. It has been argued that learning a human language will require the participation of the human user as teacher or mediator (Seabra Lopes and Chauhan, 2007; Steels and Kaplan, 2002; Thomaz and Breazeal, 2008). A menu-based human-robot interaction interface was designed which allows the user to perform three main teaching actions: *teach* the object's category name; *ask* the name of the object selected by the user; and provide *corrections* if the agent makes mistake in classification. Additional options are also available in this interface where the objective of these extra options is to create alternate classification scenarios, where the agent can make errors in classification, thus leading the user to provide *corrective feedback*. All the aspects of this interface are discussed in Section 3.2.

Evaluation of language grounding approaches will have to take into account the open-ended nature of word learning. However, most of the approaches in the literature are evaluated in closed setting where the set of categories is predefined. It is essential that, when evaluating such agents, explicit consideration should be given to assess the learning performance on scaling-up to larger vocabularies/categories. Existing evaluation methods do not allow such assessment. Due to the lack of suitable approaches to evaluate open-ended learning algorithms, a novel experimental evaluation methodology is proposed for open-ended word/category learning. This methodology can be useful for comparing the word learning capabilities of different agents and for assessing research progress on scaling-up to larger vocabularies.

Systematic experiments were carried out to evaluate different learning approaches at the task of category learning and vocabulary acquisition. Learning performance varied from one learning approach to another. Depending on the learning approach the agent was shown to incrementally acquire anywhere between 3 (the least number of categories learned using one-class learning architecture) to 293 categories (highest number of categories learned using the approach based on multi-classifier system). It was also shown that the most successful learning approach is capable of learning far more categories (than 293) although this could not be conclusively tested because of the limitations imposed by the experimental settings.

1.3 Thesis organization

The remainder of the thesis is organized as follows:

Chapter 2: This chapter provides additional background on relevant aspects of the problem of language learning, keeping in mind the scope of this thesis. Both theoretical and computational modeling perspectives are addressed. Finally, the related work on computational approaches to grounding human language in robotic agents is reviewed.

Chapter 3: This chapter begins by characterizing the requirements and features of open-ended learning. The following section is dedicated to the description of the interface designed for human-robot interaction which allows a human instructor to teach the robotic agent the names of the objects present in a visually shared environment. Later, a variety of characteristics are outlined which are considered essential for cognitive architectures that are designed to ground vocabulary in an open-ended way. Keeping these characteristics in mind, four different category learning architectures are proposed. These architectures differ in the approaches used for learning and classification, which are the matter of discussion in another chapter. The final section describes the physical architecture of the agent.

Chapter 4: In this chapter, multiple approaches to object representation are proposed. These representations are designed specifically to describe aspects of the shape of objects. On the one hand, feature-based representations are explored (shape signatures). On the other hand, by extracting the visual components of an object, graph-based representations are derived from the spatial arrangement of these components. Additionally, several similarity metrics are discussed, one of them being an original proposal in this work. These measures, in combination with specific decision rules and object representations, lead to multiple individual classifiers, that are integrated into a multi-classifier architecture.

Chapter 5: This chapter details the learning and classification approaches explored in this thesis, namely: simple instance-based learning with a single classifier; one-class classification using support-vector data descriptions; and instance-based learning with multiple classifiers and meta-learning. Additionally, for the approaches based on instance-based learning (which are memory intensive), a simple forgetting mechanism is also proposed which attempts to minimize memory usage while trying to maintain the classification performance.

Chapter 6: This chapter describes the approach for grounding spoken words. The chapter begins by discussing the approach used for extracting features from spoken words. Later, the method used for representing words and "word categories", as well as a similarity measure designed to compare these representations, are described. Finally, a novel approach to learning and classification is proposed where word categories lead to dynamic formation and reorganization of object categories, while word categories themselves are formed and reorganized through clustering of word instances.

Chapter 7: This chapter is dedicated to classical evaluation of various individual classifiers which were designed to take advantage of the relation between specific instance representations, similarity measures and classification rules. In particular, these classifiers are evaluated using k-fold cross-validation at the task of one-step as well as incremental category learning.

Chapter 8: This chapter reports the experiments and discusses the results obtained through open-ended evaluation of the different learning architectures proposed in the thesis. A novel protocol for evaluating open-ended learning approaches is proposed. Using this protocol, a systematic and thorough evaluation as well as comparison of the different learning architectures and the respective underlying learning mechanisms is carried out.

Chapter 9: The final chapter presents and discusses the conclusions and points towards future research directions.

2
Grounding human language in robots: a survey

A survey of the computational models for language grounding is presented in this chapter. To get a perspective of the progress, it is essential that the discussion begins with a historical context and follows through to the state of the art. The chapter begins with a brief discussion on the theories aimed at explaining the phenomenon of language acquisition in humans. Different theories explore the role of society, culture, situatedness/embodiment as well as perceptual and memory capacities necessary for language acquisition.

The theoretical approaches are at the root of several computational models of language emergence, evolution and acquisition. Primarily, these models have been exploited for testing/verifying the advantages and limitations of different theoretical approaches. More recently, however, there has been a focus on practical applicability, especially in the field of robotics where researchers draw ideas from these theories for designing agents that can communicate with and acquire language from humans and other agents. The core of the chapter presents and reviews the key computational models, which have appeared in literature over the years, as well as provide a survey of the state of the art.

2.1 Theoretical approaches to language emergence and acquisition

In two influential works, Noam Chomsky reasoned that a language cannot be acquired only through outside stimuli and an innate '*instinct to learn language*' is essential to explain the language acquisition process (Chomsky, 1957, 1959).

He substantiates this argument by discussing the problem of induction in syntax acquisition. That is, inductive reasoning, alone, is not sufficient to explain the rapid rate at which infants acquire their native language. Syntax of a language is usually not taught explicitly to infants. The utterances that infants are able to disambiguate from their native languages are compatible with an infinite number of structural arrangements (leading to infinite potential grammars). Completely nonsensical but grammatically correct sentences can be generated from a repertoire of words. Even if individual words in such sentences are meaningful, these sentences hold no meaning and are statistically improbable to exist in human utterances or a corpus. However, children are capable of performing the remarkable feat of acquiring the grammar of their native languages only on the basis of observations of sentences (casual observation, imitation, corrections by the verbal community etc.), that too at a remarkable speed. Chomsky reasoned that strictly probabilistic approaches that suggest that experience alone is sufficient for language acquisition (since the child speaks the language of the verbal community she is exposed to) are not sufficient to explain some of the most basic problems in language acquisition. He argued, "a refusal to study the contribution of the child to language learning permits only a superficial account of language acquisition" (Chomsky, 1959). This has led to what now is called, the "Poverty of Stimulus Hypothesis" (Chomsky, 1975; Marcus, 1999). This hypothesis states that environmental input is insufficient to express the ease and rapidity at which children acquire language, and infants must have biologically endowed language specific capabilities that facilitate language acquisition.

Chomsky's arguments were strengthened by the observations that languages of the world share many common features (*universals*) (Greenberg, 1963; Hawkins, 1990; Hockett, 1966; Mairal and Gil, 2006). Hockett (1960, 1966) outlined multiple 'design features' unique to human languages and the resulting universal traits. Although Hockett's *universals* are widely accepted, some of them have been shown to be highly probable generalizations across languages rather than absolute universals (Evans and Levinson, 2009). Greenberg and colleagues (Greenberg, 1963) argued for a different class of language *universals* that can be divided into two main categories: universal implications and statistical universals. Universal implications are the properties of a language which indicate that 'if X is present in a language, then Y will be as well'. Statistical universals make predictions, such as, 'for every language that has X , it is more likely to have Y than Z'.

The key early view that stemmed out of Chomsky's arguments and observation of universals is the *Nativist* view (Pinker, 1994). Nativists argue that the presence of universal regularities across human languages is a strong evidence that certain universal features are innately available (hard-wired) to humans and facilitate a more complex language acquisition task. According to this view, on the one hand, humans are biologically endowed with innate language specific design features (or configurational principles), which explains the universal regularities across languages. On the other hand, the structural differences across languages are explained by the 'parameters' (mental switches) that set the universal options to values specific to an infant's native language. Early nativist theories considered word-order - identified by the positions of abstract linguistic constructs (such as, *Subject-Verb-Object*) in a sentence - to be the most critical parameter (Chomsky, 1965; Gold, 1967). However, prominent nativist theories argue that these parameters are semantic in nature. Early semantic perception by infants provides the appropriate foundational material for language acquisition by allowing them to associate simple words (e.g. concrete nouns) and sentences to their meanings. By understanding these rudimentary language constructs, infants are able to form a naive theory of their native grammar by tracking the place of known words in a sentence (Gleitman and Landau, 1994) or by being able to create syntactic trees (rudimentary generalizations) from learned sentences and words (Pinker, 1984). More recently, researchers have also suggested that phonological and prosodic cues (rather than semantic ones) can also help in learning syntactic rules, even before infants have acquired any linguistic knowledge (Mehler et al., 2004; Nespor et al., 2003).

Empiricist approaches (Elman et al., 1996; Tomasello, 2000a) to language emergence and acquisition stand in contrast to the nativist stance. Empiricists argue against the presence of any innate biological endowment that is language-particular. According to this view, humans have evolved general-purpose cognitive capabilities whcih are recruited for the language learning task. These capabilities are theorized to be sufficient to extract the statistical regularities in linguistic input (Elman, 2004; Elman et al., 1996). Recent studies, however, show that statistical learning alone is not sufficient to explain the complete language acquisition process. Other mechanisms (e.g. rule-base generalization) are essential even at the most basic level of word segmentation (Peña et al., 2002; Toro et al., 2008). Toro et al. (2008) observed, for example, not all linguistic representations are equally suitable for statistical learning. In experiments on artificial grammars, they noticed that vowels were preferred for rule-based generalizations and consonants for statistical learning mechanisms.

As mentioned earlier, the presence of universal cross-linguistic regularities is widely accepted. However, the existence of universality across languages has recently been questioned (Evans and Levinson, 2009). They argue that the universal nature of languages has not been substantiated by strong experimental evidence. Many of the suggested universals are rather trite generalizations than absolute universals. The languages where striking similarities have been found share cultural-historical origins. They argue that language is a socio-cultural product where innate (evolved) human capabilities set functional and cognitive constraints but do not dictate linguistic structures (see also Cowley, 2007; Love, 2004; Pagel, 2009). Strong influence of society and culture (human-human interaction and cooperation) on language acquisition has been previously suggested by (Tomasello, 2000a,b). Tomasello argued that social cognitive functions (like imitation, intention/emotion/goal recognition), that evolved independently from language, along with statistical learning, play the key role in successful [cultural] transfer (and acquisition) of language.

The last 25 years have seen a lot of research directed towards identifying the role of society and culture on language learning. In the same period, many researchers have increasingly argued for the role of physical body in language acquisition. Independent of whether there is a biological 'instinct to learn language' or not, there is no doubt that language is transferred from a verbal community to an infant through social interactions. Such interactions can be implicit (e.g. casual observation and imitation of the verbal community an infant is exposed to) or explicit (e.g. corrective feedback from the verbal community). Social interactions occur in a physical world (*i.e.* these interactions are *situated* in reality) and are the "linguistic source" from which infants have to build a mental theory of their environment language. This implies that the perception of these interactions has to be constrained by the sensory-motor information that is available to the infant. *Embodied* theories hypothesize that sensory-motor information is an essential content of semantic representations (Barsalou, 1999; Clark, 1997; Harnad, 1990; Jackendoff, 2002). This hypothesis was proposed as a reply to an old question - how do words refer to entities of the world (Harnad, 1990; Newell, 1980; Searle, 1980)? Early theories of language acquisition (both nativist and empiricist) did not focus on this problem. Most of their effort was spent on the structural aspects (*i.e.* syntax and grammar) of language. Words (or word representations), here, are purely symbolic and had no correspondence with the items they refer to. In contrast, embodied theories argue that the meaning of a word lies in its association with the entity of the world it refers to and is embodied in our interactions with the world. The dominant view in this scenario is: for a word to have a meaning it has to be grounded in sensory-motor representations (Barsalou, 1999; Barsalou et al., 2003; Clark, 1997; Jackendoff, 2002; Meteyard and Vigliocco, 2008). Embodied theories are also at the core of the bio-cultural and social theories of language acquisition mentioned before. Here embodiment is considered pertinent for situated interaction and for grounding language in sensory-motor experience.

It is generally accepted that infants have innate capabilities that help in language acquisition. However, the extent to which these capabilities are specific to humans (pre-existing cognitive abilities and constraints utilized for acquiring language) or to language (principles encoded in an innate language faculty), is a matter of hot debate (Hauser, 2000; Pinker and Jackendoff, 2005; Tomasello, 2003). Recently, researchers (see Fitch, 2011; Fitch et al., 2005; Hauser et al., 2002) have argued for an integrative stance where mechanisms for language acquisition are divided into two sets: 'faculty of language in narrow sense' (FLN) and 'faculty of language in broad sense' (FLB). FLN consists of mechanisms that are unique to humans and to language. FLB, on the other hand, covers general-purpose cognitive elements such as, embodied sensory-motor perception, memory systems, social cognition etc. Hauser et al. (2002) exemplified this by suggesting that acquisition of complex syntax is uniquely human and FLB mechanisms alone are insufficient to explain this phenomenon. Both, FLN and FLB, are considered essential for language acquisition. FLB utilizes domain-general statistical learning and various constraints are set by human perception, socio-cultural cognition, categorical perception, memory systems, to assist language acquisition. These FLB functionalities are neither specific to language nor unique to humans. The set of mechanisms that constitute FLN are considered to be very few (Hauser et al. (2002) suggested that FLN only includes recursion). Together, FLN and FLB mechanisms are considered sufficient for language acquisition.

Various aspects of the theories discussed above have been substantiated empirically by evidence coming from several fields of research (psycholinguistics, linguistic typology, developmental psychology, behavioral science, cognitive science, evolutionary biology and more recently neuroscience). In the past twenty years, computational modeling has also played an increasingly important role as a test platform for many of these theories (and their combinations). Dictated by the scope of this thesis, further discussion will be limited to these computational models and the role they have played (and are playing) in advancing our understanding of processes involved in language emergence and acquisition.

2.2 Early computational models of language acquisition

Early language modeling efforts focused only on the acquisition of syntax. These models were a direct consequence of Chomsky's arguments (outlined in the beginning of the previous section) and attempt to resolve the induction problem proposed by Chomsky. Gold (1967) framed syntax acquisition as a computational problem. He suggested that new linguistic instances (*i.e.* sentences) initially get mapped to one of a possibly infinite set of grammars. As more instances are introduced, the mapping function modifies its behavior and gradually (through hypothesis elimination) converges towards the correct grammar. Gold's model had a great influence over the design of early computational models of syntax acquisition. The focus of early models was on designing appropriate mapping/learning functions that assist in convergence towards the grammar based on grammatically correct input sentences. On the other hand, some models hypothesize that the number of potential grammars available are much smaller (not infinite), thus reducing the size of the set of potential grammars from which the grammar of the given language has to be found.

Nativists argued that the constraints set by the *parameters* encoded in innate linguistic *principles* restrict what can be considered a legal grammar, leading to a much reduced size of potential grammars to be mapped (Chomsky, 1965). Principles, here, are believed to be the universal features common to all languages and parameters are the language variables that characterize the native language. The configurational parameters used in these models are structural (identified by the word order). Positive examples (*i.e.* grammatically correct sentences) that clearly identify properties which trigger one or more parameters, set those parameters. A grammar of a given input language gets converged when all the parameters have been set. In general, many nativist language modeling efforts have followed this approach (see Dresher and Kaye, 1990; Fodor, 1998; Gibson and Wexler, 1994).

In the nativist models, current parameter values and a correct utterance are the sole inputs used to identify the next parameter to be set. That is, these models have no memory. Grammar convergence decision is based solely on the current state of the model. *Empiricists* reason that if erroneous/ambiguous input gets introduced to a nativist model, at best, it will converge to a wrong grammar, and at worst, not converge at all. Empirical models are based on the assumption that *parameter* setting should be based on empirical evidence. Once enough utterances have triggered a particular parameter, only then it is set. Statistical regularities identified in the stored utterances are used to set the parameters probabilistically. The grammar that is identified by the parameters with highest probability is taken as the current grammar of the input language. As the number of available sentences increases, the likelihood of converging to the correct grammar also increases. The probabilistic nature of these approaches makes them immune to the presence of a few erroneous sentences. Works by Briscoe (1999), Kapur and Clark (1994) and Yang (2002) are some of the representative empirical models of grammar acquisition.

The nativist and empirical approaches discussed above have only been shown to be successful on very simplistic artificial grammars. None of these (and similar) models, to date, have been successful at learning anything close to a natural human language. The main reason, it has been argued, is that these models are generalizing syntactic rules only on the basis of the linguistic input (Pinker, 1984). In these models, words are symbols inside a computer and, it is the computational processes operating over these symbols that lead to a grammar. Such formal symbol systems were theorized in AI literature by Simon and Newell as powerful models of cognition (Newell and Simon, 1972; Simon, 1979). Formal symbol systems consider symbol formation and symbol manipulation as two separate and independent processes. Irrespective of how a symbol is formed, the focus of approaches based on formal symbol systems is to create models of symbol manipulation. Here, the interesting (and important) property of a symbol was not what it refers to, but how it can be manipulated.

These symbols, however, are parasitic and lack semantic content (Pinker, 1984; Pylyshyn, 1985; Rumelhart, 1979). Their meanings lie inside the head of the person interpreting them and/or the algorithm designer, but not inside the computer manipulating these symbols (Harnad, 1990). The models of language acquisition that do not take "meanings" into account lack cognitive plausibility. Pinker (1984), for instance, argued that the *cues* which help grammar construction are not structural (like word order used in the models discussed above) but semantic in nature. He reasoned that early acquisition of meanings of simple words and sentences is essential for building syntax.

As opposed to the purely symbolic approaches, an initial solution to the problem of associating semantic content to the linguistic input was proposed in the connectionist literature. Especially Elman (1990, 1991, 2004), in successive publications, strongly argued that *connectionist* models of language acquisition are suitable for imparting semantic content to a word (see also Elman et al., 1996). He reasoned that the stream of words in an utterance is similar to any other sensory stimuli that act directly on mental states. Activation of hidden units in a trained connectionist network, in response to an input word, is considered analogous to the activations of mental states. Elman (2004) argued that the phonological, syntactic and semantic properties of a word are revealed by the effects the word has on those mental states. However, such connectionist approaches, just like early nativist and empirical approaches, fail to address one key problem (as mentioned in previous subsection): how do words (or word representations) refer to entities of the world?

As aspects related to perception and sensory-motor control were largely overlooked, establishing the connection between symbols and their referents remained an open issue. The problem of making the semantic interpretation of a formal symbol system intrinsic to that system was called "the symbol grounding problem" (Harnad, 1990). Many of the recent computational models for language learning are based on grounding linguistic symbols, especially vocabulary. A review of these models will be the focus of the next section.

2.3 Computational approaches for language grounding

Both reasoning and language processing involve manipulation of symbols. However, these symbolic representations are amodal in the sense that they have no obvious correspondence or resemblance to their referents (Barsalou, 1999; Harnad, 1990). This limitation of classical symbolic AI led to a vigorous reaction, generally known as "situated" or "embodied" AI and, in particular, to the "intelligence without representation" views of Brooks (1990, 1991). Central to the theory of embodiment is the hypothesis that bodily experiences (*i.e.* sensory-motor perception and action) are a necessary part of semantic representations and processing (Meteyard et al., 2012). The models discussed in this section detail the robotic prototypes that were designed to explicitly ground human language in their sensori-motor perception.

The symbol grounding problem was originally formulated as a problem of formal symbol systems and classical AI (Harnad, 1990). Most research on symbol grounding has been taking place within cognitive science, usually with a strong cognitive modeling flavor and, therefore, with concerns for psychological plausibility (see survey of Cangelosi, 2005). However, it is becoming necessary to study symbol and language grounding from an engineering perspective, that is, having in mind the development of machines with reasoning and language skills suitable for practical applications. The main criterion here is no longer the psychological plausibility of the approaches but their utility. This is consistent with a modern view of AI, which no longer focuses on solving problems by simulating human intelligence, but rather on developing practically useful systems with the most suitable approaches.

Motivations behind designing computational systems for language learning (evolution, emergence and acquisition) are twofold: *cognitive modeling*; and *practical applicability* (Seabra Lopes and Chauhan, 2007; Steels, 2003). From the scientific perspective, computational models are designed as a framework for comprehensive evaluation of the hypotheses proposed by various cognitive theories (outlined in Section 2.1). On the other hand, the practical applicability perspective draws ideas from these theories in order to build (and test) novel computational approaches and mechanisms to engineer artificial agents (robotic/software) that can communicate with humans and other agents.

Models are analytical tools. Given a predefined set of assumptions,these models act as a (limited) simulation of reality. These presumptions greatly impact the analysis of these models and a poor choice can void any plausibility of such models as a reflection of the theory being tested. It has been argued that such models are "valuable to the degree that they explicitly illustrate the consequences of the set of assumptions they embody" (Batali, 1998) . Whereas, practical systems are engineering solutions and their relevance is identified by their success at the task at hand.

Focus of this thesis is on mechanisms for word learning. This is a basic language acquisition task that relies on external symbol grounding mechanisms. For artificial agents, the problem is designing suitable mechanisms for this. Increasingly, over the past 25 years, many research groups have proposed (and developed) a multitude of cognitive models and robotic prototypes to model human language acquisition in embodied agents placed in social settings. Much of these efforts have been on developing robotic agents that acquire a series of words or labels for naming certain categories of objects. In addition to grounding vocabulary, in recent years, multiple groups have begun directing research efforts towards grounding syntax as well.

The twofold distinction between cognitive and practical applicability perspectives is based solely on their respective functional objectives. The following discussion ignores this distinction and is structured to give a general overview of a variety of computational approaches proposed, over the years, for language grounding.

2.3.1 Grounding language in visual perception

Early computational models of symbol (specifically, words) grounding were primarily developed using connectionist approaches (see the survey of Cangelosi, 2005). The resurgence of connectionism in the 1980's led various authors to propose hybrid symbolic/connectionist approaches. In particular, Harnad (1990) proposed a hybrid approach to the "symbol grounding problem", which consists of grounding bottom-up symbolic representations in iconic representations (sensory projections of objects) and categorical representations (learned or innate connectionist functions capable of extracting invariant features from sensory projections).

Elementary symbols are the names of these categories. More complex representations are obtained by aggregating elementary symbols.

Harnad, Hanson, and Lubin (1991, 1995) studied categorical perception effects (within-category compression and between-category expansion) with a three-layer feed-forward network. The work involved the sorting of lines into three categories ("short", "medium", "long"). Plunkett and collaborators (Plunkett and Sinha, 1992; Plunkett et al., 1992) use a dual-route connectionist architecture with auto-associative learning for studying language production and understanding. Retinal and verbal information were present in both input and output layers, and the network had two hidden layers. After training, the network could be used both for language generation (object category name, given visual perception) and understanding (object visualization given the name). Sales and Evans (1995) used a dual-route architecture based on "weightless artificial neurons". They claim that their system can easily acquire 50 grounded nouns, although the demonstration is limited to three object categories ("apple", "jar" and "cup").

Greco, Riga, and Cangelosi (2003) study grounding transfer, that is, the process of building composed symbolic representations from grounded elementary symbols, as originally proposed by Cangelosi and Harnad (2000). In this work, the connectionist networks are trained in three stages. The first two stages learn the basic categories (color, shape and/or texture) and corresponding names. Explicit corrective feedback is provided in case of error in learning. Once the names are grounded in category representations, in the final stage, new names and categories are acquired solely from the previously grounded knowledge. They present two simulations: one with eight basic categories (four shape categories and four texture categories) leading to grounding of four composed categories, and the other with six basic categories (three color categories and three shape categories) leading to nine composed categories. In these simulations, the networks were shown to successfully ground the names of the basic categories and acquire new higher-order composed categories using only symbolic descriptions.

Steels (2003); Steels and Kaplan (2002) use the notion of "language game" to develop a social learning framework through which an AIBO robot can learn its first words with human mediation. The mediator, as a teacher, points to objects and provides their names. The robot uses color histograms and an instance-based learning method to learn word meanings. The mediator can also ask questions and provide feedback on the robot's answers. Names were learned for three objects: "Poo-Chi", "Red Ball" and "Smiley". With concrete robotic experiments, Steels and Kaplan show that unsupervised category formation may produce categories that are completely unrelated to the categories which are needed for grounding the words of the used language. They therefore conclude that social interaction must be used to help the learner focus on what needs to be learned. This is in line with previous linguistic and philosophical theories, including the Sapir-Whorf thesis (Talmy, 2000; Yoshida and Smith, 2005).

Levinson, Squire, Lin, and McClain (2005) describe a robot that learns to associate meanings using a cascade of hidden Markov models, where this cascade is a combination of the auditory, visual and concept (combined visual and auditory information) sub-models. After about 30 minutes of training, the robot is able to associate linguistic expressions with four objects: a green ball, a red ball, a toy dog and a toy cat. The linguistic expressions designate two abstract categories ("animal" and "ball") and four concrete categories ("green ball", "red ball", "dog" and "cat").

Roy and Pentland (Roy, 2003, 2005; Roy and Pentland, 2002) presented a system that learns to segment words out of continuous speech from a caregiver while associating these wordswith co-occurring visual categories. The implementation assumes that caregivers tend to repeat words referring to salient objects in the environment. Therefore, the system searches for recurring words in similar visual contexts. Their proposed model, CELL (Cross-channel Early Lexical Learning), discovers relevant linguistic units in speech input as well as relevant visual categories. A Hidden Markov Model (HMM) is generated from the phoneme sequence predicted for a given speech segment, where each phoneme is assigned an HMM state. The HMM state transitions are strictly left-to-right and the transition probabilities are given by the phoneme models previously trained on a context-independent dataset. To compare two speech segments, they proposed a distance metric which computes the likelihood of producing one speech segment given the HMM of the other speech segment. For the visual input, two-dimensional histograms of multiple views of an object are used for representing that object. Chi-squared distance metric was used for comparing objects. Co-occurring visual categories and linguistic units are paired together to form the so-called AV-events (Audio-Visual events). AV-events are consolidated in memory through clustering leading to lexical items. For evaluation, the authors collected utterances from infant-directed speech where parents introduced novel toys (pre-selected for the experiment) to their children. The agent was provided these utterances paired with images of the corresponding referent objects. The agent was able to ground names of seven object classes (e.g., a few toy animals, a ball.) Although much of their work has focused on word-learning, in (Roy and Pentland, 2002) they presented a probabilistic approach to acquire a simple grammar from the grounded vocabulary.

Yu and Ballard (Yu, 2005; Yu and Ballard, 2004, 2007) study, through a computational model, the interaction between social cues, lexical acquisition and object classification. Their model integrates multimodal social cues with statistical learning methods to facilitate vocabulary acquisition. One of the novelties of their approach is the use of joint-attention (a non-linguistic cue) to identify the object of interest. In particular, the temporal sequence of human motion (head, eye, hand movements) was taken as cue for identifying relevant objects in a visual scene. Objects here are represented by their shape (histograms), color and texture (Gabor filters) information. After the application of PCA, Gaussian mixtures are used to cluster the instances to describe a category. These objects and the corresponding utterances (usually sentences describing the action being carried out) recorded during the period of attention are then processed to extract "word-like units" and co-occurring meanings in the visual scene. Word-like units are represented by their corresponding phonetic transcripts. Similar words are clustered together to form a word set using a hierarchical agglomerative clustering algorithm. Pairing of words and meanings is then achieved through the Expectation-Maximization algorithm. This model has been used for grounding vocabulary in artificial agents where the agent was able to ground both action-verbs and object names in perceptual input. In addition to the social cues for joint-attention derived from body movements, the latest implementation of their model also takes prosodic information into account (Yu and Ballard, 2007). This model was tested on videos of infant directed speech, where the agent was shown to learn 12 object categories and successfully associated 26 words (such as, *kitty-cat*, *meow*, *hand*, *mirror*, *bird*, *eye*, *see*) related to these object categories.

Dindo and colleagues (Chella et al., 2009; Dindo and Zambuto, 2009), in successive publications, presented another approach for grounding words and rudimentary syntax in the agent's perception. Instead of focusing on naming concrete objects (e.g. names of toys, fruits and other concrete referents), they investigated the acquisition words referring to certain generic perceptual categories (e.g. color words like *red*, *blue*, *green*; shape descriptors like *circle*, *square*; spatial terms, such as *above*; and words describing size, such as *area*). Given multiple utterances describing an object in different visual scenarios, a probabilistic similarity metric is used to cluster the visual features co-occurring with a single word. Thus forming a lexical item, that is, a word associated with the semantic category represented by the cluster. Words associated with similar (based on a threshold) semantic categories are then collected to form a syntactic category. Given the syntactic categories (and associated probabilities) and information from word-order in utterances (modeled using Markov chains), they created a deterministic parser modeled using a Finite State Automaton (FSA). In their implementation, the parser was developed only for identifying spatial relations in an utterance. For the experimental evaluation, 200 utterances from two human participants paired with 50 images of individual objects were recorded (it is not clear how many object categories were used during experiments). Each recording is a simple description of the object in a given image. Evaluation of the agent's performance is based on two tasks: object description and description understanding. In the object description task, using the FSA, the agent had to create a simple sentence to describe an image. In the description understanding task, given a human-spoken utterance, the agent had to identify the correct object amongst the given 50 object images. Overall, the agent's accuracy over these tasks and was found on an average to be 80.5% and 78.9% respectively for each task (Chella et al., 2009).

As an extension of this work, in a recent publication (Dindo and Zambuto, 2010), Dindo and Zambuto have implemented their word-learning model on Nao[1] (a commercial humanoid robotic platform). The main differences with respect to the old model are: use of joint attention, by detecting demonstrator's face, hand, direction of gaze, to locate the object of reference; and more natural human-robot interaction. In the older version of the model, for a given utterance, the referent object was directly available to the agent (very limited ambiguity in what is being referred to). In the new scenario, no such limitation is set for object description. An utterance can refer to any object present in a visual scene. In the new model, word-meaning associations are formed using information available from joint-attention and a modified probabilistic method (based on multi-instance learning approach) for creating semantic categories.

Skočaj et al. (2007, 2008) present a robotic agent which is able to incrementally learn visual categories and can associate the category descriptions to the vocabulary. The cognitive module of the agent is designed such that there is interdependence between human-robot interaction (*i.e.* protocol for teaching vocabulary) and category learning. An incremental approach to category learning is taken where only positive instances are used for representing visual categories. Individual objects are represented using simple *color*, *shape* and *size* features. Additional features, derived from relative distance between objects, are used to represent spatial information. In initial work, the categories were represented using mean and variance of instances belonging to each category (leading to a prototype) (Skočaj et al., 2007), later work mixtures of Gaussians were used (Skočaj et al., 2008). A user describes a scene (pairs of objects) using the vocabulary associated with different object features and spatial relations. From multiple scene descriptions, the agent gradually learns the visual concepts and associates the corresponding vocabulary to these concepts. In multiple experiments, the agent was shown to learn 21 visual concepts in an open-ended manner. They also investigate how different levels of participation of a human language instructor influences visual category formation. Three levels of human participation were tested:

1. Strictly supervised (tutor-driven approach): Robot performs category and vocabulary learning only when instructed by the human instructor;

2. Semi-supervised (tutor-supervised approach): Robot has independent motivation and only when the robot is unsure of the visual input it requests the instructor for correct information; and

3. Unsupervised (exploratory approach): Robot creates visual categories with no interaction with the user.

They reached the conclusion that learning should begin with the tutor-driven approach and, as the robot begins to form consistent basic concepts, interactions with the robot can be reduced

[1]http://www.aldebaran-robotics.com

to minimal interaction (semi-supervised approach). That is, in their approach, the role of a tutor (and language) in helping build basic concepts is necessary. If the basic concepts have been correctly formed, later requirement of human intervention is occasional. In addition to learning concepts related to color, size and shape, their model was also shown to successfully learn simple spatial concepts. With similar motivations, Fritz et al. (2010) have designed a mixed initiative instructor-based visual category learning system that combines supervised and unsupervised learning in a unified framework.

Kirstein and Wersing (2011); Kirstein et al. (2012) take life-long learning approach to visual category learning. Although originally not stated as a word-learning problem, their work is of direct relevance for works on vocabulary acquisition. Humans acquire vocabulary throughout their lives. Designing agents with similar capability will require us to consider mechanisms that allow grounding of vocabulary throughout their lives. Kirstein and colleagues present an online, incremental and interactive approach to life-long learning of visual categories (and associated labels/words). Their category learning vector quantization (cLVQ) architecture is an examplar-based incremental learning network, combined with category-specific forward feature selection. This allows incremental learning of new categories and online modification of existing ones. Categories are represented by very high dimensional feature vectors that contain both color and shape information. Color information is represented by RGB histograms, whereas shape is represented by feature detectors obtained by unsupervised learning based on invariant sparse coding (C2 features), and parts-based features derived from SIFT-descriptors. The learning mechanism of cLVQ involves selecting the most crucial features from a series of high dimensional feature vectors that almost exclusively belong to that specific category. In latest experiments, they showed successful learning of 15 different categories (5 color categories and 10 shape categories) (Kirstein et al., 2012).

Very few vocabulary acquisition models account for grounding homonyms. The more noticable models are the following: the model used in the language games of Nowak, Plotkin, and Krakauer (1999); a child psychology inspired early word learning model of Regier et al. (2001); and, Gold's (Gold et al., 2009) social word learning model. The first two models describe plausible associative homonym formation models, but they are not models of vocabulary acquisition. The language acquisition model of Gold et al. (2009) is based on dynamic decision trees, which can account for words with both single and multiple meanings, but (at the moment) is very limited in its vocabulary acquisition capabilities and overall performance. The system of Roy and Pentland (2002) took multiple views of objects into account when learning their names. In practice, these different views can be taken as different meanings of the learned words.

2.3.2 Beyond vision: grounding language in bodily interactions

As can be noticed from the discussion above, the focus of many word-learning models has been on grounding vocabulary predominantly in visual perception. Although this is the most common direction, some researches have developed robotic platforms that physically interact with objects and can perceive their environment through additional sensory modalities. Such works usually rely on robots that can manipulate objects in their environment. In addition to the visual input, these robotic agents can perceive information from various other sensors (e.g. tactile, force, sonar, infrared). In humans, many action words (e.g. push, pull, poke amongst others) are grounded not only in visual but also in other bodily interactions (e.g. tactile perception). Take sensations perceived through human skin as an example: touch, immediate pain from a contact injury, tickle, itching etc. Words referring to such sensations can not be grounded in visual perception alone. Mental representations of such experiences, in addition to visual perception, are derived through information amalgamation across multiple sensory-motor sensations. However, visual perception, as can be observed in the models discussed earlier, can be used to infer actions (e.g. by noticing a human user moving/holding/pointing an object). In this case, robot's internal representations are based only on passive visual observations. Active interaction with the objects allows a robot to learn the affordances of objects (*i.e.* the relation between actions and their effects) (Gibson, 1979). Embodiments that allow robotic agents to actively engage with and bodily perceive their environment to learn affordances will assist in grounding words that refer to related actions and effects. In other words, the sensory-motor apparatus that allows a robot to have access to their own actions can, for example, facilitate learning vocabulary related to (but not limited to) action words (e.g. verbs like picking, moving, touching, holding).

Many researchers, in recent years, have begun exploring complex robotic platforms (simulated and real), endowed with high degrees of freedom and a multitude of sensory-motor functionalities, for grounding language. Takamuku et al. (2006) developed a system for learning categories based on the behavior/functions of objects. The underlying assumption is that the objects sharing similar types of behavior and functions will belong to the same category. In particular, they took objects from four different categories ("ball", "box", "cylinder" and "toy car"), where each of these objects has a different rolling behavior. Robot's interaction with the object involves kicking or moving the object. The model of the object's movement (*i.e.* the object behavior) is then learned, during a set training period, using reinforcement learning. Finally a Hebbian network is used for grounding the object's name (provided by an instructor) in a combined object's behavior model and the object's visual features (color histogram) . After being trained on the four previously mentioned objects, four new objects of the same categories were introduced to the robot. Based on the interaction with the new objects, the robot was shown to successfully classify these objects.

Krunic et al. (2009) presented an affordance based vocabulary grounding model, where a humanoid torso (Baltazar) equipped with visual sensors and two multi-fingered grippers with tactile and force sensors. The affordance model of Baltazar is based on finding statistical regularities (using a Bayesian network) in co-occurring stimuli as it explores the objects in the environment (by manipulation, observation of effects of manipulation and extraction of visual features of objects) (Montesano et al., 2008). In (Krunic et al., 2009), the affordance model has been modified to integrate co-occurring spoken input. The protocol of social interaction is designed such that a human instructor describes (in a simplistic grammar) the actions being carried out by the robot. During evaluation, the robot was shown to successfully learn 12 object categories and 49 associated words. Additional sensory-motor capabilities allowed Baltazar to learn object names, action words (*touching, touched, taps, tapped, tapping* etc.) and effect words (*falling, falls, rising, rise, moving, still* etc.).

Iwahashi and colleagues (Iwahashi, 2006; Iwahashi et al., 2010; Sugiura and Iwahashi, 2007) approach the word-learning problem in an incremental and online manner. Their robotic platform consists a robotic-arm with multi-fingered gripper (with tactile sensors), a stereo-vision camera and an infrared sensor. They have developed an online learning approach (LCore) based on a Bayesian probabilistic framework that allows incremental word and category learning. The experimental set-up involves a human user describing objects and/or actions in an evironment visually shared with the robotic agent. Robot's participation can be passive (e.g. human user moves the object and describes the action/object) or active (e.g. human user asks the robot to move an object). During active participation, in case the robot makes a mistake, the user verbally corrects the robot. This interaction is carried out until the robot performs the correct action (leading to a belief about the world shared between robot and human). In a passive or a correct active interaction, robot stores object features (color, shape, size and tactile information), object movement configurations and phoneme strings (of each word) identified in a spoken utterance. LCore, using new and previously stored knowledge, calculates joint-probabilities over co-occurring speech, visual (shape, size and color descriptors) and tactile information. This process leads to grounding names of concrete categories (e.g. object names) and names of other perceptual characteristics (e.g. words referring to color, shape or size). Additionally, their agent is also able to learn motion concepts (modeled by HMMs) and is able to ground some action words (e.g. move-over, place-on).

Combining classical symbolic AI with vocabulary grounding, Connell et al. (2012) present Eli, a robotic agent that can partially reason using grounded symbols. They propose ELI (Extensible Language Interface), which allows a user to command the robot "Eli" (using a restricted subset of natural language along with a few hand gestures) to perform fetch and carry tasks, as well as, to teach the names of objects and actions (*i.e. nouns and verbs*). The robot, which is equipped with a robotic arm, a camera and a microphone, grounds the names of objects in visual perception. The approach to learn new categories is instance based. The reported experiments indicate open-ended acquisition of new words and categories (although this is not explicitly mentioned). The action words are grounded using stepwise instructions, where each instruction corresponds to a known primitive action. A new action word is then associated to a sequence of basic actions (arm movements). The taught words are later used by the robot's reasoner (designed as a Finite-State Machine) which utilizes the newly learned symbols to reason about the scene as well as it's own actions. Although much of the (preexisting) symbols used by the reasoner are not grounded, the main innovation is that the new grounded symbols can be harmoniously integrated with the existing set of symbols.

2.4 Summary

The focus of this chapter was on presenting the state of the art in computational approaches to language grounding. A brief overview of the theoretical approaches was also presented, where it was shown that recent theories propose an integrative stance where sensory-motor embodiments, cognitive capabilities, and culture and society are considered pertinent for language emergence, evolution and acquisition.

Taking inspiration from different theoretical arguments, a variety of computational models for language learning have been proposed over the years. The earliest models primarily focused on the acquisition of syntax. These models were purely symbolic, since words had no correspondence to the entities of the world they refer to. This has been a major criticism of the purely symbolic approaches, and led to a strong reaction and to divergence from symbolic AI towards embodied AI.

In the last 25 years, a variety of computational approaches have been proposed, where embodiment has taken a central position. The core of this chapter presented a state of the art survey of these approaches. Most of these approaches are implemented on robotic agents and explore vocabulary through human-agent interaction. The acquired vocabulary is then grounded in visual perception. More recently, several researchers have also begun to explore word learning with robots which can physically interact with objects and can perceive through other sensors (in addition to a visual sensors). This allows for grounding vocabulary beyond object names, and is currently an emerging area of research.

3 Agent architecture

Language acquisition is a social phenomenon where society (the language source) and the processes involved in social cognition play a central role, especially, by assisting in language transfer (Cowley, 2007; Tomasello, 2000a,b). Designing the agents that can acquire human language will require the participation of the human user as teacher or mediator, in order to transfer the language of the user to the agent (Seabra Lopes and Chauhan, 2007; Steels, 2003; Thomaz and Breazeal, 2008). Designing general-purpose, educable robots that can learn and be taught by humans is also considered at the core of developing robots with human-level artificial intelligence (Nilsson, 2005).

In addition, as a cognitive tool, the main purpose of a language is to support the communication about the entities of the world. Meaning formation, on the other hand, is a cognitive task concerned with the internal representation of these entities in an individual's "brain". This distinction is extremely important because it demonstrates that any two individuals share a language when they have the same words grounded to the same entities, independent of their respective processes of meaning formation. This is exploited here for teaching a human language to a robotic agent through human assistance. Meaning formation for a robot cannot be directly compared with that for humans, still, robots can learn a human language if they can ground the human language symbols (words) and rules (syntax) in their perception of the world (Steels, 2008).

Meaning formation is highly dependent on the representations extracted from the raw sensory-motor input and on the methods used for category learning and recognition. Since word learning is an open-ended domain, e.g. we (humans) learn and acquire words through-

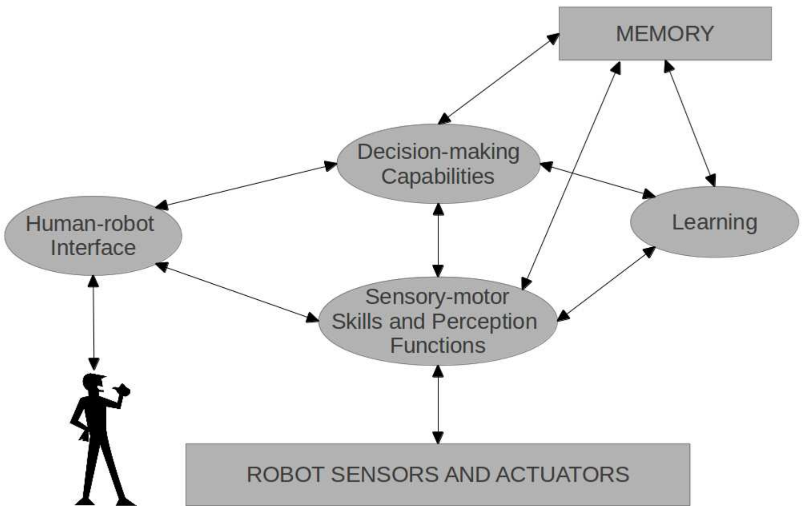

MEMORY
Decision-making Capabilities
Human-robot Interface
Learning
Sensory-motor Skills and Perception Functions
ROBOT SENSORS AND ACTUATORS

Figure 3.1: A general cognitive architecture for intelligent robots (Seabra Lopes and Teixeira, 2000). This architecture is taken as the baseline for different architectures investigated in this thesis.

---

out our lives, language acquisition must be supported by long-term learning capabilities. Moreover, the learning has to be partially supervised, incremental and run on-line, allowing the human user to teach new words and categories to the agent, at runtime and throughout agent's life. Finally, for reasoning and acting upon its world, an intelligent robotic system will have to recruit the interaction, perception and learning faculties.

All these different capabilities - human-robot communication, sensory-motor skills and perception, learning and decision-making - are inherently interdependent and are considered essential in designing intelligent robotic agents (Langley et al., 2009; Seabra Lopes and Teixeira, 2000). Seabra Lopes and Teixeira (2000) proposed a cognitive architecture that exemplifies the integrative nature of these capabilities (see Figure 3.1) and is taken as the baseline for the architectures presented in this chapter.

To design agents that can acquire human vocabulary and ground words in their sensory-motor perception, from the discussion until now, it can be observed that the following four key aspects will have to be taken into consideration:

1. Sensory-motor perception: sensory-motor embodiments to perceive and act on the world, and functionalities for extracting appropriate (complementary and informative) representations from the raw sensor input;

2. Human-robot interaction: a communication interface that facilitates language transfer from a human user (or a language proficient agent) to the robotic agent;

3. Learning: Mechanisms that allow incremental and open-ended learning.

4. Decision-making: for a robotic agent to be considered intelligent, it is essential that it has capabilities for reasoning and decision making.

To support the experimental work in this thesis, a simple agent was developed, where the agent's cognitive architecture integrated these four correlated aspects into a single framework. Since the focus of this thesis is specifically on designing mechanisms that allow an agent to ground vocabulary, the aspects related to decision-making are limited to classification decisions. The physical architecture of this agent, presented in the Section 3.5, includes a camera and a robotic-arm as the primary sensory-motor embodiments for perceiving and acting upon its environment.

At the most basic level, the human-robot communication interface implemented for this agent (Section 3.2) allows a human user to show objects in their visually shared scene and teach the names of these objects to the agent. In addition to teaching, the interface also allows the user to request simple actions and provide corrective feedback to the agent (further facilitating language transfer).

Four different category learning architecture implementations, presented in Sections 3.3 and 3.4, were explored on this agent. These architectures were designed to support on-line, incremental and open-ended category learning. The main difference between these implementations is the learning and classification approach. In the simplest case (Section 3.3.1), the learning approach instance-based. In the second case (Section 3.3.2), a one-class learning approach, based on the support-vector data descriptions (SVDD) of Tax (2001), is explored. For the last two architectures, presented in Sections 3.3.3 and 3.4), category learning approach is instance-based, but the classification decisions are supported by multiple classifiers and classifier combinations. These two architectures also include a meta-learning component. In the first three architectures, the human instructor provides names of objects as text strings. In contrast, the final architecture grounds spoken words.

Keeping in mind the open-ended nature of word learning, the discussion in this chapter begins by outlining the requirements and features of open-ended learning in the following Section.

3.1 Characteristics of incremental and open-ended learning

Language grounding is highly dependent on the techniques and methods being used for learning. Open-ended domains, like, category learning and language acquisition, must be supported by long-term learning and adaptation capabilities. For that purpose, the learning system of the agent should exhibit several basic properties (as proposed by Seabra Lopes and Wang (2002)), namely:

- *Supervised* - to include the human instructor in the learning process. This is an essential property for supporting the external/social component of symbol grounding.

- *On-line* - so that learning takes place while the agent is running.
- *Opportunistic* - the system must be prepared to accept a new example when it is observed or becomes available, rather than at predefined times or according to a predefined training schedule. This is another essential property for complying with the dynamics underlying external grounding.
- *Incremental* - it is able to adjust the learned descriptions when a new example is observed.
- *Concurrent* - it is able to handle multiple learning problems at the same time.
- *Meta-learning* - it is able to determine which learning parameters are more promising for different problems, ensuring each problem is handled effectively.

Learning methods used in most of the approaches to language grounding, as surveyed in the previous chapter (Section 2.3), do not satisfy some of these requirements. The focus of these computational models has been on developing grounding methodologies, leading to successful association of words with their corresponding sensory-motor representations. Learning methods explored in most of these models are not designed for scaling up to larger sets of words and categories and do not support on-line open-ended learning. However, few researchers have investigated agent architectures supporting open-ended category learning and language acquisition. Works of Connell et al. (2012), Skočaj et al. (2007, 2008), Iwahashi and colleagues (Iwahashi, 2006; Iwahashi et al., 2010; Sugiura and Iwahashi, 2007), Kirstein et al (Kirstein and Wersing, 2011; Kirstein et al., 2012) and one of our previous works (Seabra Lopes and Chauhan, 2007) were specifically designed for open-ended learning of visual categories. However, in their respective evaluations, none of these works were shown to scale-up to significant number of categories (highest being 21 categories reported in (Skočaj et al., 2008)). Several authors have pointed out the need for scaling up the number of acquired categories in language acquisition and symbol grounding systems (Cangelosi, 2005; Cangelosi and Harnad, 2000; Seabra Lopes and Chauhan, 2007; Steels, 2003).

As can be noticed, current approaches to the problem, although quite different from each other, all seem to be limited in the number of categories that can be learned. This limitation seems also to affect traditional incremental/life-long learning systems not specifically developed for category learning or symbol grounding, such as Learn++ (Polikar et al., 2001) and EBNN (Thrun, 1996). The instance-based learning system of Aha et al. (1991), although incremental, was demonstrated only on closed domains, *i.e.* domains with a predefined set of categories. In the only domain with more than 10 categories, actually a domain with 22 categories, best accuracy results were under 40%.

Within the field of computer vision, there is recent progress towards systems able to learn larger numbers of categories. The main works are being evaluated on Caltech-101, a well-known database composed of 8677 images of objects of 101 different categories. Recognition accuracies achieved on this problem using 15 training images per category are between 50 and 60% (Grauman and Darrell, 2007). Competitive results have also been reported in literature (for a recent example see (Uray et al., 2009)) on other common multi-class datasets, such as, COIL-100 (Nene et al., 1996) and ALOI1000 (Geusebroek et al., 2005). More recently, larger and much more challenging datasets, such as ImageNet (Deng et al., 2009), are gaining popularity in the computer vision literature where the state of the art research has achieved accuracy of up to 16% on almost 20,000 object categories (Le et al., 2012). However, all works evaluated on these datasets follow the train and test approach, rather than focusing on interactive agents with on-line learning capabilities.

In the context of classical symbolic artificial intelligence (AI), the issue of long-term learning remains largely an open issue. Attempts to apply classical AI learning techniques (explanation-based learning, case-based reasoning) in the long run have faced computational performance problems (Francis and Ram, 1993; Minton, 1990; Mooney, 1989). The most common explanation for such problems is related to the cost of testing the applicability of the acquired knowledge (production rules, cases, etc.) to concrete problems. As problem-solving time increases with the number of learned structures, there is a trade-off between utility and cost. Long-term learning is usually also included at the core of cognitive theories. (Kennedy and Trafton, 2007) investigated the support provided to long-term learning by two well-known cognitive architectures, namely Soar and ACT-R. They reported computational performance problems in both systems, namely an increase in problem-solving time, as learning continues. In the case of ACT-R, analyzed in more detail, one of the problems also identified is the inability to handle smoothly a finite and limited memory capacity.

Given the open-ended nature of category learning, different learning architectures, presented in this thesis, were designed to support online concurrent/opportunistic learning of an arbitrary set of categories. Open-endedness implies online and incremental learning, therefore some of the characteristics of incremental algorithms are directly applicable to open-ended learning algorithms. Schlimmer and Fisher (1986) proposed three main dimensions for evaluating incremental learners: the *number of observations* required by a learning system to obtain 'stable' category descriptions; the *cost of updating memory* to accommodate an observed object; and the *quality of category descriptions*. Syed et al. (1999) highlighted that these dimensions are not sufficient to quantify whether an incremental learner is able to recover, in next learning steps, when a category description begins to drift off from the 'real' category, with the introduction of new training data. This drifting off, known in the literature as *concept drift* (Schlimmer and Granger, 1986; Widmer and Kubat, 1996), occurs as the models trained on old data become inconsistent with the new incoming data. The concept drift can be *real* (e.g. the underlying distribution of a category changes over a period of time) or *perceived* (e.g. when the new training data is very noisy or an aberration that is not typical of the 'actual' category). In an incremental learning scenario, by definition, not all training data is available beforehand and, as the new data arrives in incremental steps, the learner should be able to handle concept drift. Syed et al. (1999) proposed the following requirements to avoid concept drift:

1. Stability: The prediction accuracy on the test set should not vary wildly at every incremental learning step;
2. Improvement: There should be improvement in the prediction accuracy as the training progresses and the learning algorithm sees more training examples; and
3. Recoverability: The learning method should be able to recover from its errors, that is, even if the performance drops at a certain learning step, the algorithm should be able to recover to the previous best performance.

Standard incremental learning algorithms are designed to allow the introduction of new data specifically for previously known categories. In contrast, an open-ended learning scenario implies that the agent can be exposed, at any arbitrary point in time, to new examples from previously seen as well as new categories. The introduction of new categories can significantly interfere with the existing category descriptions. Moreover, robots and software agents are limited in their perceptual abilities and, therefore, cannot learn arbitrarily large numbers of categories, particularly when perception does not enable the detection of small between-category differences. As the number of categories increases, the learning performance will evolve with phases of performance degradation (caused by either perceived or real concept drift) followed by recovery, but eventually reach a point where the learning agent is no longer able to accommodate newer categories. Seabra Lopes and Chauhan (2007) states these features of an open-ended learning process:

1. **Evolution**: Depends on the ability of the learner to adjust category representations taking into account new instances while no new categories are introduced. In this case, the introduction of new instances should lead to a gradual improvement in the prediction accuracy. Overall, on a test set, the learner's performance should also not vary wildly. Evolution, here, incorporates the "Stability" and "Improvement" criteria of Syed et al. (1999).
2. **Recovery**: The discrimination performance will generally deteriorate with the introduction of a new category. The time spent in system evolution until correcting and adjusting all current categories defines recovery. Recovery is based on classification errors and corresponding corrections as new instances become available. This feature incorporates "Improvement" and "Recoverability" criteria of Syed et al. (1999).
3. **Breakpoint**: Inability of the learner to recover and evolve when a new category is introduced.

3.2 Human-robot interaction for vocabulary acquisition

The adopted HRI (Human-Robot Interaction) approach was primarily designed to facilitate vertical transmission of words from a human instructor to the robotic agent. Vertical transmission refers to a social language transfer scenario where a set of agents acquire their language through interactions with a previous generation or from an independent population that is already language proficient. In the present scenario, the role of language proficient agent is taken by a human user, that teaches vocabulary to the robotic agent. This approach is consistent with the view that learning a human language in artificial agents will require participation of humans as language instructors (Seabra Lopes and Chauhan, 2007; Steels, 2003; Thomaz and Breazeal, 2008). This view underlies an increasing trend in recent years to develop agents that acquire language through interactions with humans (see the literature review in Section 2.3).

In the present work, a human user, acting as an instructor, teaches the robot the names of the objects present in their visually shared environment. Object names are then grounded by the robot in sensor-based object descriptions, leading to a vocabulary shared with its instructor. At any moment in time, the shared scene can contain zero or more objects and the user can change the content of the scene at her discretion by adding or removing objects.

One key problem in this scenario is: when the instructor provides the object name, how can the agent disambiguate which object (amongst several in the scene) is the user referring to. This problem surfaces when the name is not grounded in object's perception. A simple example would be, when a language learner (e.g. a child) hears a new (previously unknown) word she has to identify its referent from multiple possible hypotheses that are perceptually available to her. Resolving this ambiguity is essential if words are to be correctly grounded by the agent.

Having mentioned this, the focus of this thesis is on developing mechanisms for grounding vocabulary and simplifying assumptions were made such that the problem of word-to-referent uncertainty does not arise in the current case. More specifically, the resolution of this problem is part of the HRI interface design. The interface allows the human user to point (by mouse clicking) at the desired object. It should be noted that the assumption of non-ambiguous pointing is not unique to our case. Resolving word-to-referent ambiguity, in embodied agents that ground language through interactions with other agents (humans or otherwise), is an open problem. Much of the research on designing such agents make multiple simplifying pragmatic assumptions (especially with regards to the experimental setup) such that there is either no uncertainty in the word and its referent, or the referent of a word can be easily figured out in the scene (this is true for most of the literature surveyed in the previous chapter).

The selection of an object from the scene enables *joint attention* between the instructor and the agent. Once joint attention is established, the instructor can interact with the agent using a simplified interface. The interface provides a set of actions that the user can request the agent to perform. The primary objective, here, is to allow a user to teach names of objects, present in a shared scene, to the agent. Central to this interface is the possibility to provide *corrective feedback*. That is, in cases where the agent performs a requested action incorrectly, the user has the facility to give correction. It has been argued in literature that corrective feedback plays a significant role in early language acquisition (Tomasello, 2000a,b), especially to explain fast word-to-meaning mapping in early language learners (Carey, 1978). Additionally, the role of *joint attention* and *corrective feedback* has also been investigated in societies of artificial agents. In a set of experiments over varying populations of agents, Vogt and Coumans (2003) showed that *joint attention* and *corrective feedback* mechanisms help a population converge considerably faster towards a common vocabulary than in absence of these mechanisms.

For an agent that is designed to perform long-term learning and adaptation for acquiring vocabulary in an on-line and incremental manner, the human-agent interaction interface must allow the human user to teach vocabulary; create scenarios where corrections might be required; and provide corrective feedback. Therefore, at the most basic level of interaction, the interface allows the user to perform four main actions:

1. *Point* to select the desired object;
2. *Teach* the category name of the selected object;
3. *Ask* the category name of the selected object, which the agent will predict based on previously learned knowledge;
4. *Correct*: if the category predicted in the previous case is incorrect, the user can teach the correct category;

The robot responds by either running the relevant learning functionalities (in response to *teach* and *correct* actions) or performing classification (in response to an *ask* action by the user).

Two implementations of this basic interface were developed: one where the user provides object category names as textual input (typed directly via the keyboard) and another that supports vocal input (*i.e.* object category names are spoken). In the latter case, the robot receives verbal *teach* and *correct* instructions and responds to questions from the human user, by recording and reproducing the audio information given through teach and correct actions (recordings of words). In both approaches, the pointing action is supported as simple mouse-clicking on an image of the scene on the computer screen.

As is evident from the base-level interactions, misclassification from the agent invokes *corrective feedback* from the user. That is, inaccurate classification is central for identifying opportunities to provide feedback. Additionally, taking advantage of the sensory-motor capabilities of the agent, richer scenarios can be designed where the user's requests indirectly imply classification. In particular, the user can request the agent to interact in its environment where, unlike explicit request to classify (the base-level *ask* action), classification becomes implicit for carrying out these requests. From the mistakes of the agent at task execution, the user can infer inaccuracy in classification and provide correction. With this objective, the interface allows the user to request the agent to perform the following actions[1] (continued from the base-level interaction list):

5. Request the robot to *pick* the selected object using the robotic arm;
6. Provide a category name and request the robot to *locate* an instance of that category;
7. Provide a category name and request the robot to *locate-and-pick* an instance of that category using the arm;
8. Once an object has been picked, select a location and request the robot to *place* the object there (by clicking at the desired location on the camera frame).

To perform the actions requested using options 5-6, the agent has to classify each object present in the scene. Incorrect classification will lead the agent to choose/pick an incorrect object from the scene. In case a wrong object is chosen by the agent, user can choose option 3 to provide the correct category name of the that object.

Although different responses of the agent to user requests have been discussed above, for convenience the agent's possible responses are listed below:

- *Learning*: create new or modify existing category descriptions, and create association between the category name and its corresponding category description (response to actions 2 and 4) ;
- *Linguistic response*: provide category of the selected object back to the user (response to action 3);
- *Visual response*: visually report the results of the "locate" request (response to action 6); and
- *Manipulation actions*: the robotic arm manipulates the objects in the robot's environment (response to actions 5, 7 or 8).

Here, learning process is internal to the agent and the user has no direct method to know whether the agent successfully learned an object category. The remaining responses of the agent are visually available to the user (e.g. user can see which object was moved by the robotic-arm) and divergence from expected response leads the user to provide corrections.

[1]For these interactions, current implementation only supports textual input.

3.3 Architectures for visual category learning

In this thesis, three learning architectures were investigated which were designed to satisfy most of the basic properties listed in Section 3.1. These architectures differ in their complexity and performance based on the methods used for learning and classification. The following architectures were implemented and tested over the course of this thesis:

1. Instance-based learning with a single classifier;
2. One-class learning; and
3. Instance-based learning with multiple classifiers and meta-learning.

By organizing the categories and instances according to user's feedback, these architectures behave in a supervised way. They are online because they are integrated in the normal activity of the agent. These architectures are incremental and opportunistic since they are able to adjust categories when new instances are observed rather than requiring that training instances be given in a training phase or according to a predefined training schedule. Each of these architectures will be discussed in detail in the subsequent subsections. An architecture for grounding spoken words was also designed and implemented, as will be discussed in Section 3.4.

3.3.1 Instance-based learning with a single classifier

Instance-based learning (IBL) (Aha et al., 1991) is perhaps the simplest of the learning approaches that can fulfill the basic requirements (listed at the beginning of Section 3.3) for open-ended learning. This is a type of *lazy learning* approach (Aha, 1997), where categories are simply described by the instances (stored in the memory) that belong to that category. The primary advantage of this family of supervised learning algorithms is that the target learning function (e.g. a category description) is approximated locally. For IBL, this means that "similar" instances are considered to belong to the same category, leading to a local bias when classifying novel instances (Aha et al., 1991). Moreover, since categories are described solely by the instances belonging to that category, this approach can easily deal with multi-class situations (different categories are described by instances belonging those specific categories), as well as accommodate to changing hypothesis space (e.g. by storing more instances belonging to that category).

The similarity function (to assess similarity between two instances) and the classification rules, derived from the similarities of a novel object to the instances belonging to the different categories, are at the core of this learning approach.

Figure 3.2 provides an illustration of the architecture. When the user introduces a new object and provides its true category, by performing either *teach* or *correct* actions, the

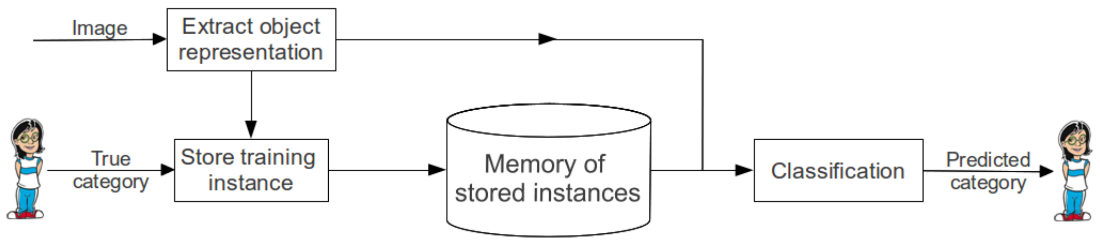


Figure 3.2: Instance-based category learning.

object's representation is simply stored in the memory along with the corresponding category label. This storage strategy accounts for both, new instances belonging to an existing category and instances belonging to a previously unknown category.

3.3.2 One-class learning

Symbol grounding involves finding the invariant perceptual properties of the objects or categories to which symbols refer (Barsalou, 1999; Harnad, 1990). This suggests that learning of symbol meanings should be (predominantly) based on positive examples. Learning from positive examples is the basis for the *one-class learning* paradigm (Japkowicz, 1999; Tax, 2001), which was adopted for one of the learning architectures.

One-class learning is an interesting candidate learning paradigm for such an open-ended domain as word learning, since it is not easy to provide counter-examples. Tax (2001) describes and experimentally compares a large number of methods from the perspective of one-class learning, including Parzen density estimator, autoassociators, SVDD (Support Vector Data Description), LVQ (Linear Vector Quantization), PCA (Principal Component Analysis), SOM (Self Organizing Maps), k-means and k-centers. One of the preferred methods is SVDD, a method that shares its foundations with the support vector classifier (Vapnik, 1995). It shows one of the best performances and is particularly good at avoiding overfitting (Tax, 2001). In addition, the evaluation time is very small. Therefore, one of the learning and classification approaches explored in this thesis is based on SVDD.

In its original form, SVDD (briefly described in Section 5.3.2), is neither incremental, nor designed for multi-class scenarios. For open-ended domains like vocabulary acquisition and category learning, the learning process needs to be incremental, online and open-ended. Therefore, in earlier work (Seabra Lopes and Chauhan, 2006, 2007), a one-class lifelong learning architecture (OCLL), was designed which supports open-ended learning (see Figure 3.3).

OCLL uses the original implementation of SVDD algorithm[2], which comes with tools for optimizing SVDD parameters. In the normal case, SVDD is trained only with positive instances (inliers) of the target class. However, both inliers and outliers can be used for finding

[2]Available with Tax's PRTools package.

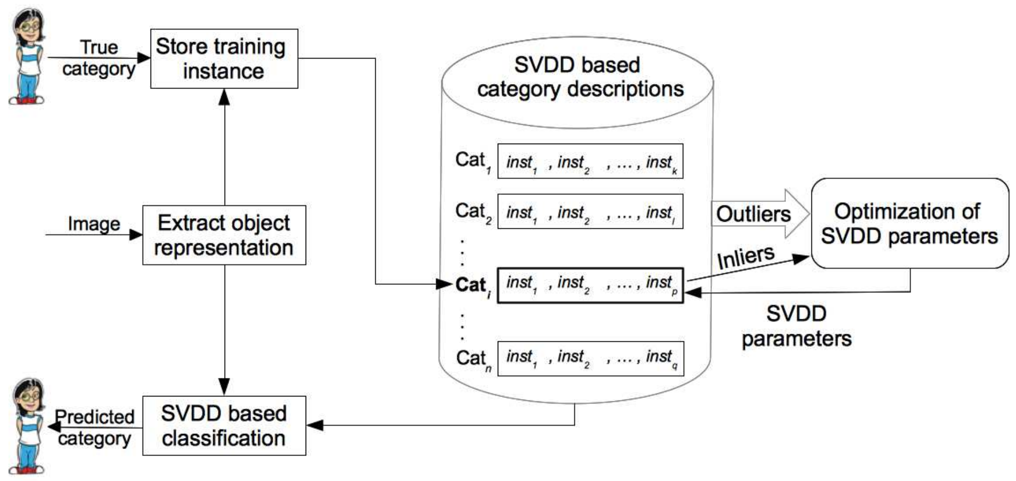

True category
Store training instance
SVDD based category descriptions
Cat$_1$ inst$_1$, inst$_2$, ..., inst$_k$
Cat$_2$ inst$_1$, inst$_2$, ..., inst$_l$
Image
Extract object representation
Outliers
Optimization of SVDD parameters
Inliers
Cat$_i$ inst$_1$, inst$_2$, ..., inst$_p$
SVDD parameters
Cat$_n$ inst$_1$, inst$_2$, ..., inst$_q$
Predicted category
SVDD based classification

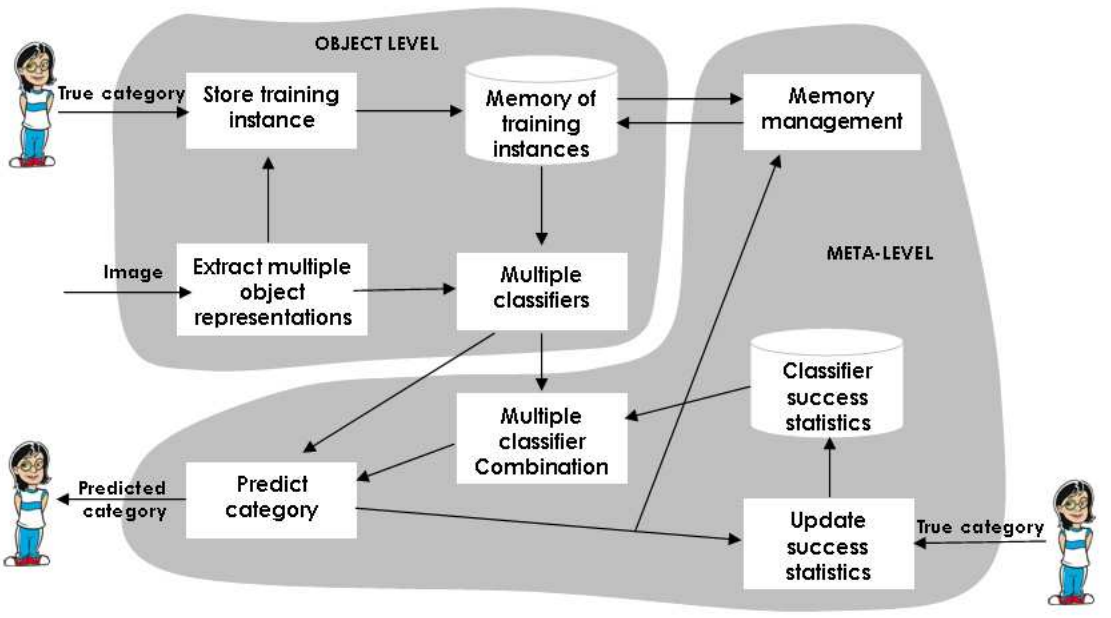
Figure 3.3: Old SVDD based category learning architecture (Seabra Lopes and Chauhan, 2007).

---

the optimal parameters. Tax (2001) had reported that the presence of a few outliers can help the optimization process to achieve tighter class boundary descriptions (or hyperspheres). In OCLL, outliers (negative instances misclassified as belonging to the target class) were used in this boundary optimization.

As mentioned previously learning is incremental and supervised, and the user is explicitly included in the language acquisition process. When an object is misclassified, the instructor has the option of providing the correct class, so that class descriptions can be improved. Given a correction from the user, OCLL will identify and retrain the class descriptions needing correction. Specifically, OCLL will add the misclassified object as outlier for retraining the classes whose hyperspheres contain the object. A standard quadratic optimization algorithm then tries to form a hypersphere (target class description) around the data by finding a set of support vectors. These support vectors are data points on the boundary of a hypersphere whose center is also determined through optimization. Experimental evaluation of OCLL showed that, although the agent was able to incrementally learn multiple categories, in multiple experiments, it was unable to learn more than 11 categories. Therefore, building upon previous work, a new architecture was designed where the key novelty, with respect to OCLL, is the optimization process of the SVDD learning parameters.

In the new approach, a genetic algorithm is used for optimization. The choice of genetic algorithm for optimizing SVDD parameters is based on the results reported in (Tavakkoli et al., 2007), where genetic optimization, in a non-incremental scenario, was shown to clearly outperform other standard optimization techniques. The new learning method and the optimization process are described in Section 5.3.



OBJECT LEVEL
META-LEVEL
True category
Store training instance
Memory of training instances
Memory management
Image
Extract multiple object representations
Multiple classifiers
Classifier success statistics
Multiple classifier Combination
Predicted category
Predict category
Update success statistics
True category

Figure 3.4: Category learning architecture supporting multiple classifier systems and meta-learning.

3.3.3 Instance-based learning with multiple classifiers and meta-learning

The previously described instance-based approach to open-ended learning was extended to use multiple feature spaces for object representation, multiple classifiers and classifier combinations. Adequate classification relies to some extent on a metacognitive self-monitoring and control loop. All learning computations are carried out during the normal execution of the agent, which allows continuous monitoring of the performance of the different classifiers. The measured classification successes of the base classifiers are used to reconfigure dynamically some of the classifier combinations as well as to select the classifier that will be used to predict the category of new unseen objects. In the proposed architecture (Figure 3.4) the metacognitive component is also concerned with memory management.

The term 'metacognition' was coined in psychology to refer to the phenomenon of cognition about cognition (Flavell, 1971). The general information-processing framework of Nelson and Narens (1990) is well known for its explicit consideration of metacognitive processes. Basically, this framework divides cognitive processes into object-level processes and meta-level processes. A so-called monitoring flow of information from the object-level to the meta-level allows the meta-level to keep a dynamic model of the object-level. Based on this model, the meta-level can send a control flow of information to the object-level, modifying which object-level behaviors are initiated and terminated.

Researchers have been moving towards the conclusion that human category learning relies on multiple memory systems and multiple representations (Ashby and O'Brien, 2005; Kruschke, 2005). Attentional selection, *i.e.* a mechanism of focusing on specific features or representations based on recent experience, has also recently been emphasized (Kruschke, 2005). These developments too suggest there is a meta-level associated to human category learning.

A lot of work in the AI field can also be seen as the interaction between object-level and meta-level processes. Cox (2005) presents a detailed survey of metacognition, in both psychology and artificial intelligence, and with the emphasis on problem-solving and story understanding tasks. Although some caveats are pointed out, metacognition is here considered an essential component for systems addressing such tasks.

The proposed learning architecture is based on the idea that using multiple representations, multiple classifiers and multiple classifier combinations, all potentially complementary of each other, can enhance global performance. Some of these ideas, particularly the use of classifier combinations, are not new in the machine learning literature (Xu et al., 1992). The main innovation in this architecture is that those complementarities are explored in an on-line learning architecture, and a simple form of meta-learning takes advantage of the on-line nature of the learning process to improve global performance. Teaching and corrective feedback actions from the human mediator are used to monitor the classification success of the individual classifiers. The measured classification successes of the individual classifiers are used to reconfigure dynamically some of the classifier combinations, and may also support the selection of the classifier that will predict the category of a new unseen object. Thus, the whole system is clearly divided into two main components. The object-level component extracts object representations, stores instances in memory and runs base classifiers. The meta-level component optimizes memory usage, monitors classifier performance, configures and runs classifier combinations and produces the category prediction for the target object.

3.4 Architecture for grounding spoken words

The architectures in the previous section focused on grounding textually communicated words. The final architecture in this chapter, presented in Figure 3.5, extends that functionality to support grounding of spoken words. More specifically, the learning architecture supporting multiple classifiers and meta-learning (Section 3.3.3) is taken as the foundation for the new architecture.

Using speech introduces new problems. The first involves the representation of object and word categories in memory. Similar to the previous architecture, an instance-based representation scheme is adopted such that an object category is represented by its known set of instances and a word category is described by the set of known word instances. Finally, corresponding object and word categories are connected to each other such that each object

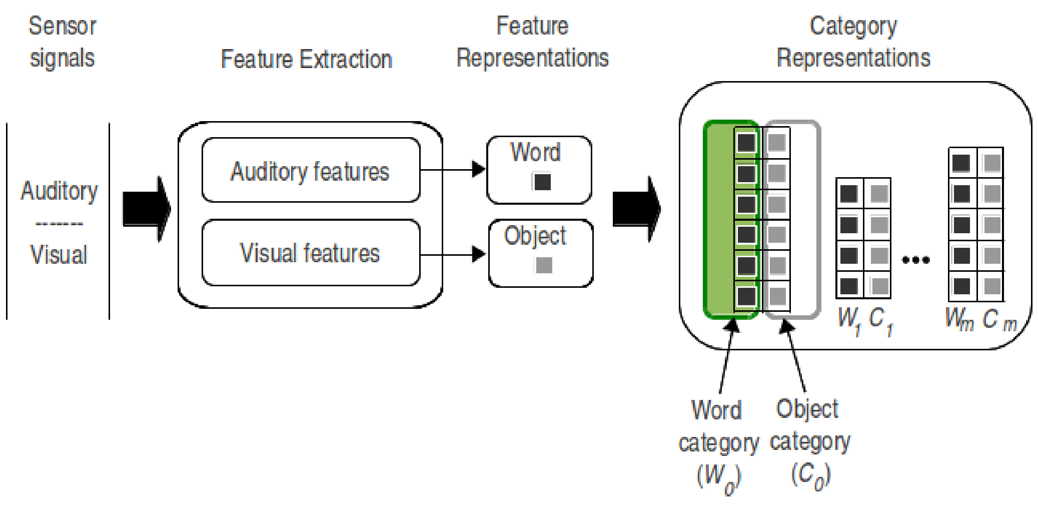


Figure 3.5: Signal perception and representation schema for grounding spoken words.

---

instance is associated to a specific word instance. When the user teaches the name of an object, the association between the representations extracted from the object image and the spoken word is always maintained.

Secondly, unlike textual input (where there is no ambiguity), when the user utters the category name of a selected object, the agent first has to classify the spoken word, that is, map the speech input to a previously known word category. Since different utterances of the same word display some amount of variation, this may lead to misclassification. In this architecture, we assume that most of the word categories are reasonably homogeneous when compared to the object categories. Therefore, word categories are formed using unsupervised clustering over the perceived spoken words. The objective here is that, as the number of word instances for a given category increases, similar instances will gradually cluster together, leading to a faithful representation of that word category. The clustering routine is run on a particular word category description, each time the user performs a *teaching* or a *correction* action. Since each object instance is associated to a specific word instance, each clustering action will lead to dynamically form and reorganize object categories. A complete description of different aspects of this architecture is provided in Chapter 6.

The object level and the meta-level components - that entail object classification using multiple classifiers and classifier combinations - have not been modified in this architecture and are exactly the same as described for the original multi-classifiers and meta-learning architecture (see Figure 3.4).

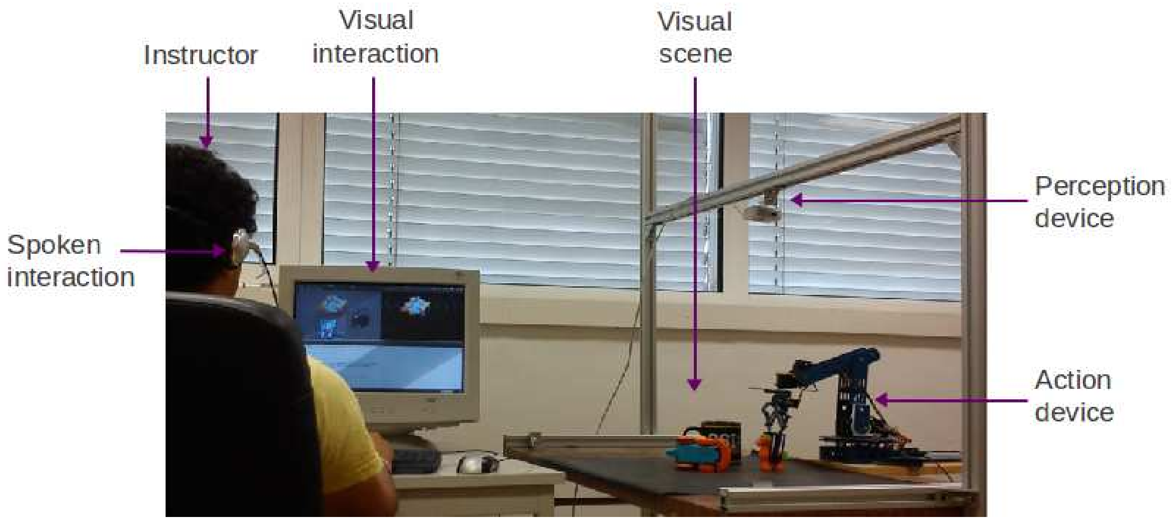


Figure 3.6: Experimental scenario

3.5 Physical architecture of the developed agent

A simple physical agent was designed to support the experimental work in this thesis (see Figure 3.6 for the complete experimental setup). This agent is physically embodied with a robotic arm and a camera, and all its cognitive functions are carried out on an attached computer. The agent's environment includes a human user (normally not visible to the agent), a dynamic scenario (a table top) visually shared with the user and objects whose names the user may wish to teach.

To physically interact with its environment, the agent is endowed with the capability to manipulate objects using a robotic arm which is mounted on the center of one of the long edges of the table top. This arm is an SG6-UT[3] educational arm manufactured by Crust Crawler Robotics. It is shipped with a PSC-USB board (Parallax Servo Controller with USB interface) and contains 6 servos for 5 arm joints and a gripper, comprising 6 degrees of freedom. The pay load of this arm is around 400 grams. A client interface was developed for the PSC-USB enabling to control the arm directly from the computer.

As reported in literature (see survey in the previous chapter), much of the work on vocabulary acquisition has been explored in visual domain. Taking a similar approach, this thesis also explores grounding of vocabulary in visual perception. The agent's primary perception device is an IEEE1394 compliant digital camera[4]. The camera is placed in a fixed position above the table such that the area of the table within the range of the robotic arm is visually

[3]SG6-UT - a 6 degree of freedom robotic arm, supplied on-line at:
http://www.crustcrawler.com/

[4]IEEE 1394 compliant firwire camera called "fire-i", supplied on-line by Unibrain at:
http://www.unibrain.com/Products/VisionImg/Fire_i_DC.htm

accessible to the agent. The table top was chosen to be black in color to simplify object detection and extraction from the camera frame. A linear mapping between the camera and the robotic-arm coordinates is assumed. Appendix A presents the details on robotic-arm control to manipulate objects that have been extracted from the camera frame.

The computer runs the human-robot interaction interface, the perception module, learning and recognition module and robotic arm control functions.

The HRI interface allows the human user to point at the desired object. The pointing action is supported by mouse clicking at the object in the robot's visual scene (camera frame). Using a region growing algorithm, the object is then extracted from the background. That is, the instructor provides the name for the object that has previously been selected by her and therefore the agent does not have to disambiguate amongst multiple possible referents.

Two modes of communication are supported between the agent and a human user. In one mode, interaction between a human user and the agent is allowed only through textual input (text typed directly into a computer terminal). This mode exemplifies the cases where no ambiguity is expected in communication between the agent and the human user. In the second mode, the agent supports spoken input. That is, the reliability associated with the communication is lost. In the later mode, the agent is embodied with a headset microphone (Genius HS-02N) to perceive vocal signals. This microphone was configured to record single channel acoustic signals at a sample rate of 16 kHz with 16 bit resolution.

3.6 Summary

One of the key aspects outlined in this chapter are the capabilities necessary for designing the architecture of an embodied agent for it to be able to ground human vocabulary in an open-ended way. The characteristics of long-term, incremental and open-ended learning were discussed. It was argued that open-ended category learning processes go through stages of evolution and recovery, and eventually may reach a breakpoint. As the work incorporates the human user as mediator of the learning process, a set of basic instructor actions was identified: point, teach, ask and correct.

In general, four key inter-dependent capabilities are considered essential for intelligent service robots: perceptual and sensory-motor skills, decision-making capabilities, human-robot interaction and appropriate learning mechanisms. A simple robotic-agent was developed which incorporates all of these capabilities. This agent is designed to acquire concrete nouns through interaction with a human teacher. The agent is supported by a camera as its primary perception sensor, and the words taught by the teacher are grounded in visual perception through visual category learning. In addition to teaching object categories, the user can also request the agent to interact with its environment (using search, locate, pick and place commands).

Several open-ended category learning architectures were outlined. Three of them were designed for visual category learning when there is no ambiguity in the input word from the instructor: instance-based learning with a single classifier (Section 3.3.1); one-class learning with support-vector data descriptions (Section 3.3.2); and instance-based learning with multiple classifiers and meta-learning (Sections 3.3.3). One final architecture was designed for grounding spoken words (Section 3.4). In this case, mechanisms were designed for learning word as well as visual categories. These mechanisms were then combined such that the word categories get associated correctly with their corresponding visual categories.

4
Representations and similarity measures

A robot's access to the environment is determined by its sensory-motor devices. These devices are the primary sources of perception. This is consistent with the theories of embodied/situated cognition, where, cognition is considered to be directly governed by the interactions of the body with the environment. For the raw data to "make sense", it must be processed for extracting the most relevant information. This processing involves noise reduction and extraction of smaller, more informative, representations of the data. The characteristics of the representations are dependent on the problem at hand and are generally defined a priori.

As mentioned in the previous chapter, a video camera is the principal sensing device being used with the our robotic agent. This device allows visual perception by continuously capturing raw data from the visible scene and transmitting it to the post-sensory modules of the agent's cognitive architecture. For any object present in the robot's visual scene, the agent is able to build internal representations of that object. Methods were developed to extract multiple object representations, where, the focus was on capturing the shape information.

In the field of computer vision, a multitude of methods has been proposed for shape representation. These methods broadly fall into two major categories (Zhang and Lu, 2004): region-based and contour-based. The main difference between these two categories is the amount of information (extracted from an object image) that is taken into account for deriving shape representations. Region-based methods use all the pixels present in the original object image, whereas, contour-based methods derive shape information only from the edge pixels.

Both approaches were explored in this thesis and several novel representations are proposed for describing the shape of an object. On the one hand, multiple concise and computationally inexpensive representations (in the form of feature vectors) are extracted from an object's contour. On the other hand, using color information of all the pixels in the image, the main regions or components are extracted and, the object's shape is represented by a graph derived from the spatial arrangement of these components. In Section 4.1, different approaches to object representation are explained in detail.

Feature vectors and graphs of components provide a global description of an object's shape. This class of representations has been explored by researchers over many years (Loncaric, 1998; Pavlidis, 1978; Zhang and Lu, 2004). However, local descriptors, such as SIFT (Scale-invariant Feature Transform) (Lowe, 2004) and SURF (Speeded Up Robust Feature) (Bay et al., 2008) have gained increasing popularity in recent years. As we observe, local descriptors perform quite well for specific categories (e.g. bottle of a specific brand of beer), but tend to perform poorly with general categories (e.g. any kind of bottle) (Antunes and Lopes, 2013; Pereira and Seabra Lopes, 2009). Moreover, 'bag of features' approaches, where features can be SIFT and SURF features, provide little structural information about objects, which can be seen as a disadvantage from the point of view of language acquisition (Roy, 2005).

Each of these representations (whether contour-based or region-based) is designed to provide a description of some aspect of the shape of an object. A good representation should distinguish/discriminate between instances belonging to different categories, at least to a certain extent. That is, in principle, representations of instances belonging to the same category should be more *similar* to each other than to the representations of the instances of other categories.

To compare object representations, at the most basic level, a variety of similarity metrics were used. Some of these metrics are standard (*Euclidean Similarity* and *Pyramid Match*), while others were developed within the framework of this thesis (*Manhattan Pyramid Match* and *Graph of components similarity*). Most of these similarity measures are generic and can be used with any feature-based representation, whereas, for graph-based representation, described in 4.1.2, a specific metric was developed. Section 4.2 introduces the different measures of similarity explored in this thesis.

4.1 Object representations

When the user points the mouse to an object in the scene image and selects an action to take (*teach, ask* or *correct*), an edge-based counterpart of the whole image is generated using the Canny edge detector (Figure 4.1ab). From this edges image, the boundary of the object is extracted taking into account the user pointed position[1], and assuming that different objects do not occlude each other in the image. Given the boundary of the object, an edge-

[1]This is performed using a region growing algorithm

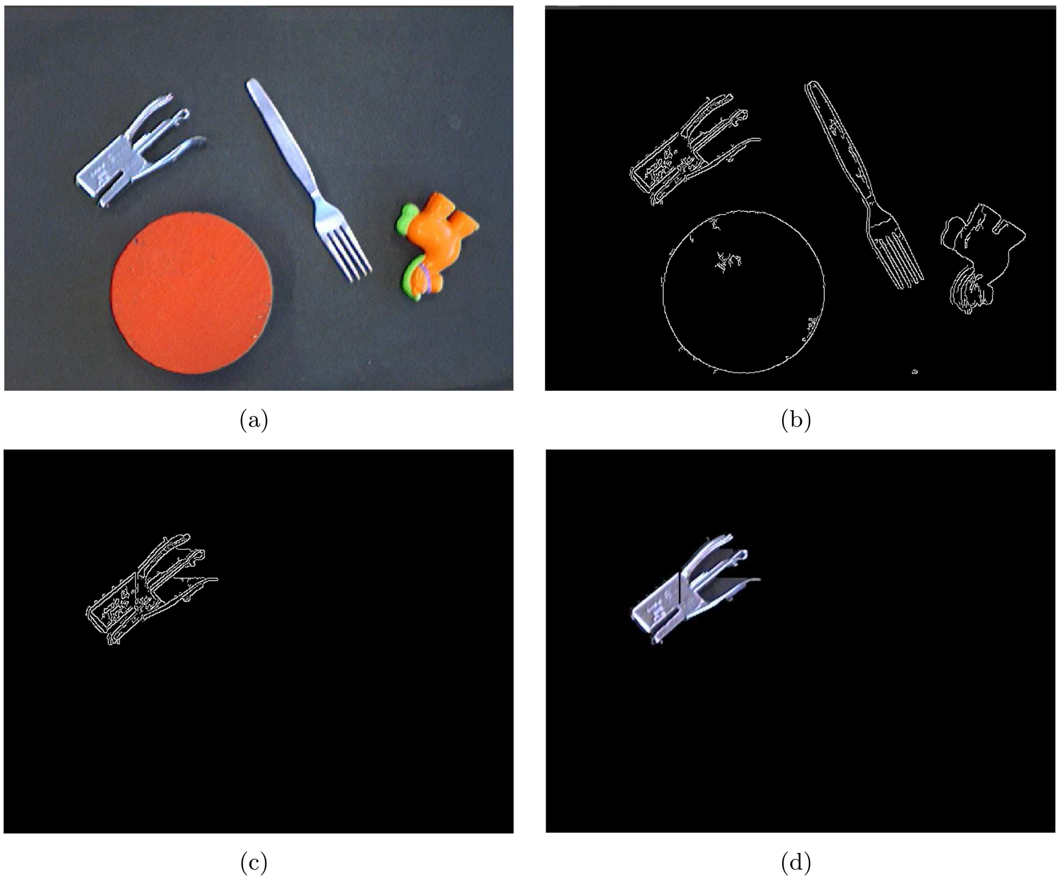


Figure 4.1: (a) A visual scene with four objects; (b) Edge-based counterpart of the visual scene; (c) Edges of the object selected by the human instructor ("*Stapler*"); (d) "*Stapler*" extracted from the original scene.

---

based image of the object is extracted from the full scene image (Figure 4.1c). Several shape signatures are extracted from the edges image of the object (Section 4.1.1).

By applying a mask derived from the edges image of the object over the original image, a color image of the object is extracted (Figure 4.1d). This image is used to derive a component-based representation of the object (Section 4.1.2).

4.1.1 Feature-based object representations

Objects should be described to the learning and classifications algorithms in terms of a small set of informative features. A small number of features will shorten the running time for the learning algorithm. Information content of the features will strongly influence the learning performance.

In the multi-classifier approach of Section 3.3.3, several possibly complementary feature spaces are explored concurrently. Most of these feature spaces are the result of segmenting

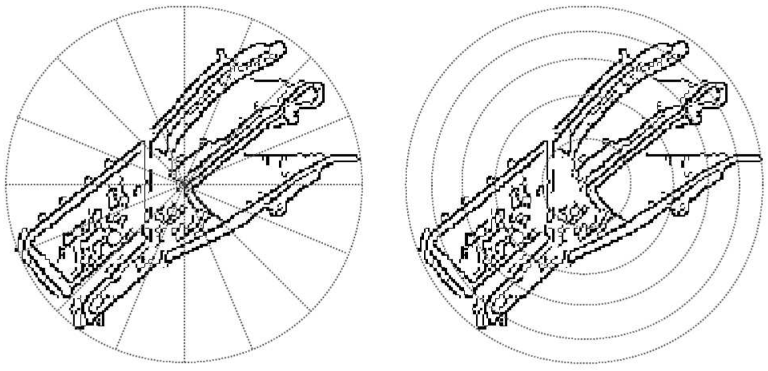
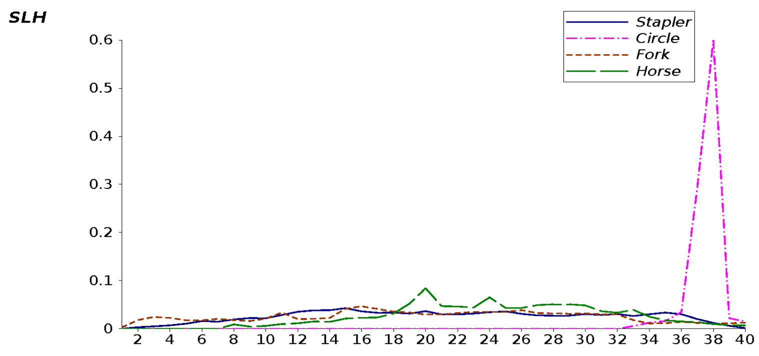


Figure 4.2: Segmentation of edges image of an object into slices (left) and layers (right)

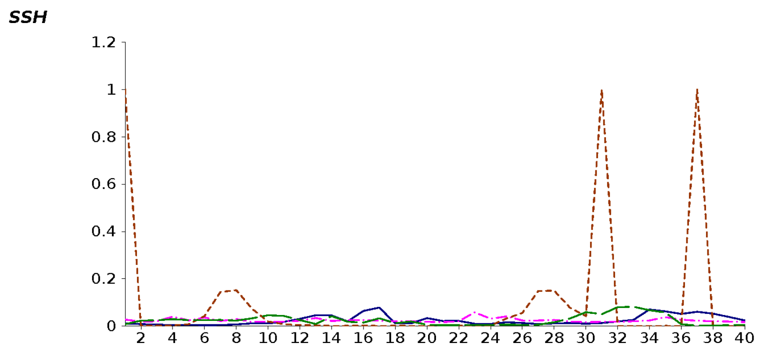
---

the smallest circle enclosing the edges image of the object. For different feature spaces, such a circle is segmented either into a number of slices (Figure 4.2, left) or a number of concentric layers (Figure 4.2, right). Current implementation uses 40 slices and 40 layers. Feature spaces based on this kind of segmentation are aimed at capturing shape information. In the following, these feature spaces are briefly described:

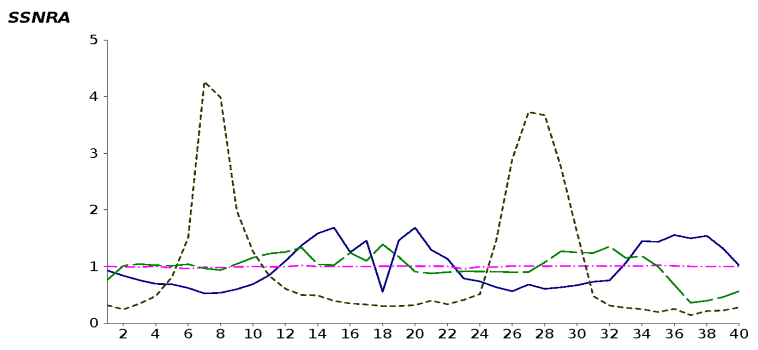
- *Shape Layers Histogram* (*SLH*): The histogram contains, for each layer, the percentage of edge pixels with respect to the total number of edge pixels of the object. This feature space is both scale-invariant and rotation-invariant. An example is given in Figure 4.3a for the four objects shown in Figure 4.1.
- *Shape Slices Histogram* (*SSH*): The histogram contains, for each slice, the percentage of edge pixels in that slice with respect to the total number of edge pixels of the object. An example is given in Figure 4.3b.
- *Shape Slices Normalized Radii Averages* (*SSNRA*): For each slice, i, the average radius of all edge pixels in that slice, R_i , is computed. In this feature space, an object is represented by a vector $\mathbf{r} = r_1 ... r_{40}$, where $r_i = R_i/R$ and R is the average of all R_i. An example is given in Figure 4.3c. *SSNRA* is the core of the feature space used in previous work (Seabra Lopes and Chauhan, 2007) and, in addition to the multi-classifier approach of Section 3.3.3, is used with the one-class learning approach of Section 3.3.2.
- *Shape Slices Normalized Radii Standard Deviations* (*SSNRSD*): For each slice, i, the radius standard deviation of all pixels in that slice, S_i , is computed. In this feature space, an object is represented by a vector $\mathbf{s} = s_1 ... s_{40}$, where $s_i = S_i/R$ and R is the average radius as mentioned above. An example is given in Figure 4.3d.
- *Normalized Radius Standard Deviation* (*RADSD*): This is a feature space composed of a single feature. Its value is the standard deviation of the normalized radii averages,



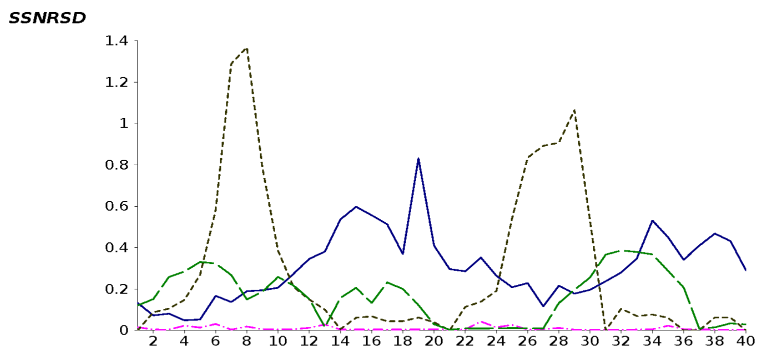
(a)



(b)



(c)



(d)

Figure 4.3: Different types of shape signatures for the four objects in Figure 4.1a: (a) Shape Layers Histogram; (b) Shape Slices Histogram; (c) Shape Slices Normalized Radii Averages; and (d) Shape Slices Normalized Radii Standard Deviations.

$\mathbf{r} = r_1 ... r_{40}$, mentioned in the previous item.

- *Area* (*AREA*): This feature space is composed of a single feature, area, defined as the total number of pixels of the object. This is the only scale-dependent feature space used in this work, and it is also the only one not providing any shape information.

In summary, the current system uses two uni-dimensional feature spaces (*AREA* and *RADSD*), and four shape-based feature spaces (*SLH, SSH, SSNRA* and *SSNRSD*). Except *AREA*, all the feature spaces are an original contribution of the work carried out in this thesis. *AREA* captures the size information. All other feature spaces are scale-invariant. The feature space based on shape layers (*SLH*) is also rotation-invariant, but not those based on slices (*SSH, SSNRSD* and *SSNRA*). For slice-based feature spaces, similarity can still be computed in a rotation-invariant way. Specifically, for feature spaces derived from shape slices, similarity between any two instances is computed as the maximum similarity between the respective feature vectors as they are circularly rotated relative to each other.

In each of the shape-based feature spaces, an object is represented by an ordered set of features that describes the shape of the object. They work as shape signatures (Zhang and Lu, 2004). The shape representations used here can be related to the so-called 'shape context' of Belongie et al. (2002). In fact, their log-polar bins are a division of the object into slices and layers. However, the shape context is not centered in the geometric center of the object. Instead, a shape context is computed for every edge pixel of the object, possibly after down-sampling. Object matching is then carried out by pairing edge pixels such that the total matching cost between histograms is minimized. So, the shape context is used as a local descriptor, whereas our feature spaces based on slices and layers are used for global object representation. Another important difference is concerned with computational complexity. While most of the steps of the similarity computation algorithm of Belongie et al. (2002) run in time quadratic to cubic in the number of edge pixels, our representations can be built and used in linear time. Our representations can be more directly compared with the Global Shape Context (Pereira and Seabra Lopes, 2009), a descriptor based on polar division of the object into slices and layers that is used as a global object descriptor.

4.1.2 Component-based object representation

The previous subsection described an approach to represent the shape of an object using feature vectors. However, feature vectors are not always considered the best representation for learning (Aha and Wettschereck, 1997). This is particularly the case when instances can be split into components leading to a structured/relational representation. Object classification based on feature-based shape representations could be better handled with relational instead of vector representations. That is also the assumption underlying the cognitive theory of recognition-by-components (Biederman, 1987). These ideas link to some of the literature on symbol systems and symbol grounding. For instance, the theory of "perceptual symbol systems" (Barsalou, 1999) emphasizes that a perceptual symbol represents a schematic component of a perception, and not a holistic experience. Also, the "Symbolic Theft Hypothesis" (Cangelosi and Harnad, 2000) emphasizes that complex categories can be learned more efficiently from more basic categories than directly form sensor data.

Based on all these considerations, in addition to the feature-vectors, a component-based/relational representation of the shape of the objects is also explored. In this case, an object is represented by its components and geometric relations between them. Generally speaking, a component is a region in an object's image with color homogeneity. Once the components have been extracted, the relations between these components are represented using non-directed graphs.

Graph of components

More specifically, an object is represented using a graph model of the spatial arrangement of its components, referred as *graph of components*. In such graph, each component is represented by a different node. The relation between a pair of components is represented by a non-directed edge connecting their respective nodes. Formally, a graph of components is described as:

$$G = < C, R, A > \tag{4.1}$$

where the three items are the following:

- $C = \{c_i\}, i = 1, ..., n_C$, is the set of all components of the object, represented by nodes in the graph;
- $R = \{(c_i, c_j)\}$ is the set of all pairs of components $c_i \in C$ and $c_j \in C$ which are in contact in the image of the object. These relations are represented as edges in the graph; and
- $A = \{(c_i, c_j, c_k)\}$, is the set of all ternary relations between components such that $(c_i, c_j) \in R, (c_i, c_k) \in R$.

Extracting the graph of components

This work uses the HSV (Hue, Saturation and Value) color space for locating and extracting object components. A list of the most relevant color ranges in the given object is initially computed: $[H_1, ..., H_n]$. Each range H_i is represented as a tuple (a, b, m, A), where a and b are respectively the start and end values (hue) of the color range, m is the most frequent color value and A is the area of the object (number of pixels) in that color range. These ranges are found using the algorithm described in (Seabra Lopes et al., 2007). Once the color ranges have been found, the image is modified by replacing all the hue values in a range H_i by its corresponding most frequent color m.

The object image is then processed such that the neighboring pixels with the same hue value are aggregated to form the initial set of components. Connections are established between any two components physically in contact with each other (*i.e.* with adjacent pixels). These components are further merged based on the following steps:

1. Compute the geometric center of each component and abstract the component as a circle with that center and the same area;
2. If the circles of any two components overlap considerably (covering more than 90% of at least one of the components) they are merged to form a single component;
3. If the circles of any two components overlap, covering an area between 35% and 90% of at least one of the components, check for hue similarity. If the hue values present in one component are similar to the ones present in the other, merge the components (if the difference in two hue values is less than 35, they are considered similar);
4. Repeat these steps until no components can be merged.

Once the merging process is complete, the contact relations between the components are determined. A contact relation between any two components c_i and c_j is found based on the following criteria:

1. If c_i and c_j are physically connected or extremely close to each other, there is a contact relation between them. More specificaly, if any two border pixels, one from each component, are closer than a threshold (3 pixels in the implementation), then the components are considered connected.
2. If a component is found to have no neighbor using the previous rule, then it is considered connected to its closest component. In this case, for simplicity, the decision is based on the distances between the borders of the circles.
3. The graph structure is derived directly from the contact relations. However, this strategy has the drawback that the graph may not always be completely connected. That is, instead of one connected graph, the graph of components may end up as a set of two or more connected subgraphs. This problem is solved by locating the pair of components with the closest contact relation (each component belongs to a separate subgraph), and then connecting these components. This procedure is repeated until there are no futher subgraphs to be connected.

Figure 4.4 displays the components and contact relations computed from an image of a "toy train" object.

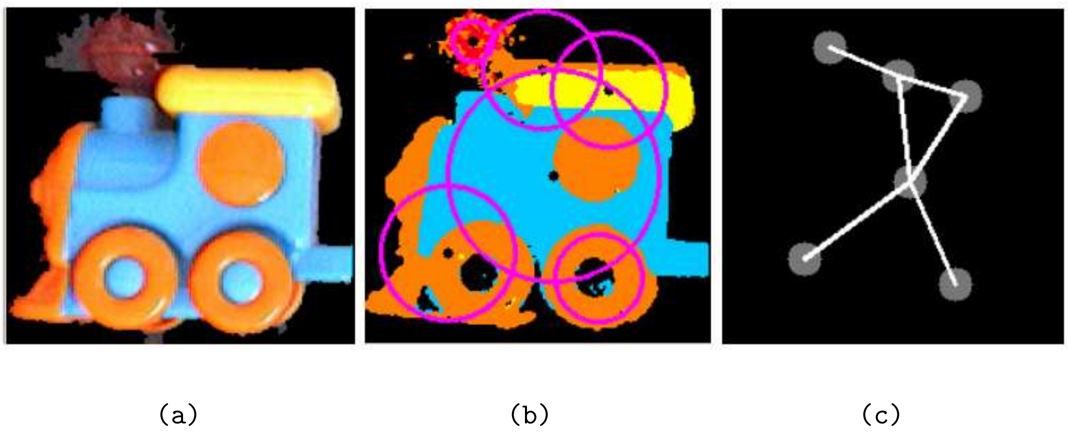


Figure 4.4: a) The “toy train” object; b) Extracted components and the circles with the same area as that of the components, centered at the geometric centers of the components (the components are extracted in HSV color space, but here, for the purpose of visualization, the second image is converted back into RGB); c) Graph of components of the given object.

Features of the graph of components

The properties of a graph of components are derived from the organization of the components inside the object. Each component c_i (*i.e.* each node of the graph) is characterized by the following features:

- a_i - relative area, given by the ratio of number of pixels in the component to the total number of pixels present in the object;
- r_i - relative radius, given by the ratio between the radius of a circle, with the same area as the component, and the radius of a circle with the same area as the whole object; and
- d_i - relative degree, given by the ratio between the degree of a node (number of neighbors) and the total number of nodes in the graph.

The contact relation $(c_i, c_j) \in R$ between two components c_i and c_j is characterized by a single feature:

- d_{ij} - relative distance between the components c_i and c_j, measured as the ratio of the distance between the geometric centers of these components and the diameter of a circle with the area of the object.

Finally, each ternary relation $(c_i, c_j, c_k) \in A$ is also characterized by a single feature:

- a_{ijk} - relative angle between edges (c_i, c_j) and (c_i, c_k), given by the ratio of the smaller angle between the two edges and π.

The graph and the described features constitute a scale, translation and rotation invariant representation of an object.

4.2 Similarity measures

For comparing object representations, several measures of similarity were explored in this thesis. For feature-based representations three similarity measures were used. First similarity metric is derived directly from simple Euclidean distance. Alternatively, two similarity metrics – Pyramid match (P_Δ) and Manhattan Pyramid Similarity (MPS) – are derived from multi-resolution matching algorithms similar to the matching algorithm used in the pyramid match kernel of Grauman and Darrell (2005, 2007). Finally, a novel similarity measure was designed to compare representations based on the *graph of components*. In this section we will elaborate on these different measures of similarity.

4.2.1 Euclidean similarity measure

As objects are represented as feature vectors in most of the feature spaces described in the previous section, an obvious similarity measure is inverse Euclidean distance. To convert a distance metric to a similarity measure, multiple approaches can be used. Two classical approaches are (Ashby and Alfonso-Reese, 1995; Kruschke, 2005): inverse of distance ($1/D$); and exponential similarity $exp(-\alpha D^\beta)$, where α and β are the constants that respectively give the slope and decay of the exponential function. In the present case case, Euclidean similarity measure (ES) is defined as the inverse of Euclidean distance. That is, the Euclidean similarity between two feature vectors $\mathbf{x}$ and $\mathbf{y}$ is given as:

$$ES(\mathbf{x}, \mathbf{y}) = 1/D(\mathbf{x}, \mathbf{y}) \tag{4.2}$$

where $D(\mathbf{x}, \mathbf{y})$ is the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$.

4.2.2 Pyramid match score

Pyramid match score ($\tilde{P}_\Delta$) is a multi-resolution similarity measure derived directly from the pyramid match kernel (PMK) (Grauman and Darrell, 2005, 2007). This kernel and its variants (e.g. *spatial pyramid match kernel* of Lazebnik et al. (2006)) have gained increasing attention, especially in the field of visual classification (Bosch et al., 2007; Dong et al., 2008; Sahbi et al., 2011; Seabra Lopes and Chauhan, 2008; Yang et al., 2009). This kernel function was designed to enable the application of kernel-based learning methods to domains where objects are represented by unordered and variable-sized sets of features, such as sets of local features in computer vision. At the core of this family of kernels is the multi-resolution matching method called the *pyramid match.* Pyramid match is a kernel function which in this thesis is interpreted as a similarity measure[2]. In this matching approach, each feature set is mapped to a histogram pyramid, *i.e.* a multi-resolution histogram preserving the individual features distinctness at the base level. Then, at each level of resolution, the histogram pyramids are matched using a weighted histogram intersection computation. In addition to being suitable for histograms (as Grauman and Darrell (2005, 2007) proposed), this matching approach is also suitable for other ordered feature spaces (Seabra Lopes and Chauhan, 2008).

The feature-based representations used in the present work, as described in Section 4.1.1, are ordered and have a constant dimension. So mapping these representations to multi-resolution pyramids is direct. Then, the same basic matching algorithm can be applied.

Given two multi-dimensional ordered feature vectors $\mathbf{x}$ and $\mathbf{y}$, a match between them is computed at increasingly coarse resolutions. The resolution of the feature vectors is reduced by half in each successive step of the algorithm. The final pyramid match score is taken as the weighted sum of the matches computed over all resolutions. The pyramid match between $\mathbf{x}$ and $\mathbf{y}$ is given by:

$$\tilde{P}_{\Delta}(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{L-1} w_i N_i(\mathbf{x}, \mathbf{y}) \tag{4.3}$$

where L is the number of pyramid layers, $w_i = 1/2^i$ is the weight of layer i and N_i measures the additional matching at layer i, as given by:

$$N_i(\mathbf{x}, \mathbf{y}) = I(F_i(\mathbf{x}), F_i(\mathbf{y})) - I(F_{i-1}(\mathbf{x}), F_{i-1}(\mathbf{y})) \tag{4.4}$$

where $F_i(\mathbf{x})$ is the feature representation of object $\mathbf{x}$ at layer i and $I()$ is an intersection function that measures the overlap of two objects as follows:

$$I(\mathbf{a}, \mathbf{b}) = \sum_{j=1}^{r} min(a_j, b_j) \tag{4.5}$$

where $\mathbf{a}$ and $\mathbf{b}$ are feature vectors of size r, and a_i is the value of the i^{th} element of $\mathbf{a}$.

Although presented here for single-dimension vectors, the extension of this algorithm to the multi-dimensional case is straight forward.

An anomaly in matching by minimization

Similarity is directly related with proximity (or inversely related with distance) (Kruschke, 2005), but this relation is not captured when matching by minimization. For example, Eu-

[2]It has been observed in machine learning literature that kernels can be interpreted as similarity measures (Balcan and Blum, 2006; Srebro, 2007).

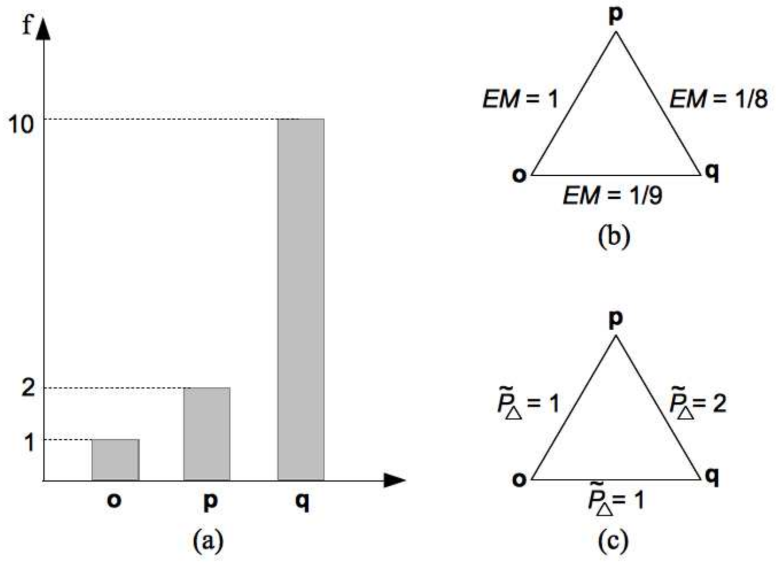

f
10
2
1
o
p
q
(a)
p
EM = 1
EM = 1/8
o
q
EM = 1/9
(b)
p
$\tilde{P}_\triangle$ = 1
$\tilde{P}_\triangle$ = 2
o
q
$\tilde{P}_\triangle$ = 1
(c)

Figure 4.5: Un-normalized Pyramid Match score ($\tilde{P}_\triangle$) anomaly: (a) Three objects represented by a single feature; (b) ES shows that object **p** is more similar to object **o**; (c) $\tilde{P}_\triangle$ would lead to consider **p** more similar to **q**.

---

clidean similarity, ES, is more robust when an object, with low feature values, is compared with other objects, with higher, but clearly different feature values. An extreme situation, for a feature space with a single feature, is illustrated in Figure 4.5: $\mathbf{o} = \{1\}$, $\mathbf{p} = \{2\}$ and $\mathbf{q} = \{10\}$.

With Euclidean similarity, we get $ES(\mathbf{o}, \mathbf{p}) = 1$, $ES(\mathbf{o}, \mathbf{q}) = 1/9$, $ES(\mathbf{p}, \mathbf{q}) = 1/8$, which correctly describes the situation, *i.e.* **o** and **p** are more similar to each other than to **q**. In contrast, with matching by minimization, the following similarities are obtained: $\tilde{P}_\triangle(\mathbf{o}, \mathbf{p}) = \tilde{P}_\triangle(\mathbf{o}, \mathbf{q}) = 1$ and $\tilde{P}_\triangle(\mathbf{p}, \mathbf{q}) = 2$. These values would suggest that **p** and **q** are more similar to each other than to **o**, which is not the case.

Normalized Pyramid Match

In the context of PMK, to avoid favoring larger input sets (which translate into histograms with larger values), Grauman and Darrell (2005, 2007) proposed to normalize the pyramid match score by the product of the self similarities of the input histograms:

$$P_\triangle(\mathbf{x}, \mathbf{y}) = \frac{\tilde{P}_\triangle(\mathbf{x}, \mathbf{y})}{\sqrt{\tilde{P}_\triangle(\mathbf{x}, \mathbf{x}).\tilde{P}_\triangle(\mathbf{y}, \mathbf{y})}} \tag{4.6}$$

This also resolves the extreme case described above. However, in borderline cases, the anomaly persists even after normalization. As an example, if the feature of the second object in the previous case, **p**, was 4, rather than 2, ES would still correctly describe the situation, *i.e.* **o** and **p** would still be more similar to each other than to **q**. However, both the unnormalized pyramid match, $\tilde{P}_\Delta$, and the normalized one, P_Δ, would lead to the conclusion that **p** and **q** were more similar to each other than to **o**.

4.2.3 Manhattan Pyramid Similarity

To combine the advantages of distance information with multi-resolution analysis in the style of PMK, we designed a new measure, the Manhattan-Pyramid Distance (MPD), which is also computed at multiple resolution scales (Chauhan and Lopes, 2012). MPD is computed by subtracting a weighted sum of distance reductions, obtained by successively lowering the resolution, from a distance computed at the base resolution level. In each step of the algorithm, resolution is divided by 2. All distances are measured using the well known Manhattan Distance (MD) measure, defined as follows:

$$MD(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{r} |x_i - y_i| \tag{4.7}$$

where $\mathbf{x}$ and $\mathbf{y}$ are feature vectors representing two objects and r is the number of features.

The distance reduction or discount at resolution level i, $R_i(\mathbf{x}, \mathbf{y})$, is given by the difference between the Manhattan distances at resolution $i-1$ and i:

$$R_i(\mathbf{x}, \mathbf{y}) = MD(F_{i-1}(\mathbf{x}), F_{i-1}(\mathbf{y})) - MD(F_i(\mathbf{x}), F_i(\mathbf{y})) \tag{4.8}$$

where $F_i(\mathbf{x})$ is the feature vector representing object $\mathbf{x}$ at resolution level i. The final discounted distance is given by:

$$MPD(\mathbf{x}, \mathbf{y}) = MD(\mathbf{x}, \mathbf{y}) - \sum_{i=1}^{L-1} w_i R_i(\mathbf{x}, \mathbf{y}) \tag{4.9}$$

where L is the number of resolution levels and $w_i = 1/2^i$ is the weight of the distance reduction at level i in the final discount.

In the experiments presented in this thesis, for more direct comparison with other similarity measures, we define the Manhattan-Pyramid similarity measure, MPS, as the inverse of MPD:

$$MPS(\mathbf{x}, \mathbf{y}) = 1/MPD(\mathbf{x}, \mathbf{y}) \tag{4.10}$$

4.2.4 Graph of components similarity measure

In addition to the representations based on feature spaces, another representation approach explored in this thesis is based on finding the components that constitute an object

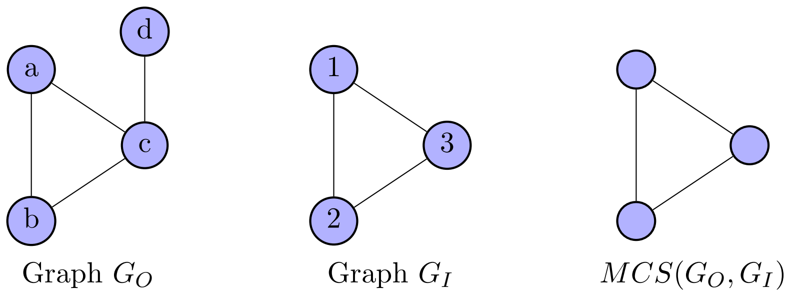


Figure 4.6: An example of $MCS(G_O, G_I)$

---

and computing a graph-based representation of the spatial relationships between these components (Section 4.1.2).

To compute a measure of similarity between two objects O and I, the graph structure similarity of their respective graphs, G_O and G_I, is evaluated. This is based on locating the Maximum Common Subgraph (MCS) between them (Figure 4.6). Cunningham et al. (2004) defines MCS as "the largest set of all linked nodes that two graphs have in common". It should be noted that MCS is not always unique. Once $MCS(G_O, G_I)$ is obtained, the following similarity measure is computed:

$$S_{MCS}(G_O, G_I) = \frac{g(MCS(G_O, G_I))}{g(G_O) + g(G_I) - g(MCS(G_O, G_I))} \tag{4.11}$$

where $g(G)$ corresponds to the sum of the total number of nodes and total number of edges of a graph G.

If $MCS(G_O, G_I)$ is not empty, it implies the existence of a common structure of nodes and edges between the graphs G_O and G_I. Given the common structure a mapping between nodes and edges of these graphs is found. This mapping is found by performing breadth first search (Diestel, 2000). Considering the Figure 4.6 again, we can see that the mapping between the nodes and edges of graphs G_O and G_I can be:

- Mapping between nodes: a↔1; b↔2; c↔3
- Mapping between edges: ab↔12; bc↔23; ca↔31

In a similar manner, the ternary relations can also be mapped. Once the $MCS(G_O, G_I)$ and the mappings for the two graphs G_O and G_I are known, their dissimilarity is measured using the following auxiliary function:

$$d_X(G_O, G_I) = \sqrt{\frac{\sum_{i=1}^{n}\left(\hat{d}_X(G_O, G_I).M_X(i) + (1 - M_X(i))\right)}{n}} \tag{4.12}$$

where

- X represents one of the three types of items in a graph of components (*i.e.* the set of components, C, the set of contact relations, R, and the set of ternary relations, A, as presented in Section 4.1.2, eqn. 4.13);
- n is the cardinality of X in G_O;
- $M_X(i) = 1$ if the i_{th} element of type X in G_O is mapped to a corresponding element in G_I, otherwise $M_X(i) = 0$;
- $\hat{d}_X(G_O, G_I) = \dfrac{\sum_{j=1}^{k} (O_X(i,j) - I_X(map(i),j))^2}{k}$
  - where $\hat{d}_X$ is the averaged Euclidean distance between the features corresponding to a specific pair of elements i, j, and ranges between 0 and 1.
  - k is the number of features for items of type X (according to previous explanations, $k = 3$ for the components in C and $k = 1$ for the elements in R and A);
  - $O_X(i,j)$ is the value of feature j of the i_{th} element of type X in G_O;
  - $I_X(i,j)$ is the value of feature j of the i_{th} element of type X in G_I;
  - $map(i)$ is the element in G_I to which element i of G_O is mapped.

$d_X(G_O, G_I)$ is the average distance for the type of item X between G_O and G_I. The distance is greatest for the elements of G_O which were not mapped to any element of G_I. In eqn. 4.12, it can be observed that for each element i which is not mapped between the two graphs, the distance is set to 1 (since $M_X(i)$ becomes 0), otherwise it is set to $\hat{d}_X(G_O, G_I)$.

Based on this, the dissimilarity between the two objects is given by a weighted average of the dissimilarities computed for the three types of items as follows:

$$D_g(G_O, G_I) = w_C d_C(G_O, G_I) + w_R d_R(G_O, G_I) + w_A d_A(G_O, G_I) \tag{4.13}$$

where w_C, w_R, and w_A are weights, with $w_C + w_R + w_A = 1$ (in the implementation $w_C = 0.55; w_R = 0.35$; and $w_A = 0.1$).

Joining the similarity measure in eqn. 4.11 and the dissimilarity measure in eqn. 4.1, the final measure of graph similarity is given as:

$$S_g(G_O, G_I) = w_S S_{MCS}(G_O, G_I) + w_d(1 - D_g(G_O, G_I)) \tag{4.14}$$

where w_S and w_d are weights (0.3 and 0.7, respectively, in the implementation).

4.3 Summary

This chapter presented multiple approaches to object representation that were designed to capture shape information (presented in Section 4.1), as well as a set of similarity measures to compare these representations (Section 4.2).

Broadly, two approaches to shape representation were presented:

1. Contour-based feature-spaces (Section 4.1.1): Multiple computationally inexpensive feature-based representations were presented. These representations are extracted from the object's contour and they mostly describe the shape of the object. Except $AREA$, all shape representations are scale-invariant. One of them (SLH) is also rotation-invariant, while the rest of the feature-spaces easily allow for rotation-invariant similarity assessment.

2. Component-based (Section 4.1.2): A component-based representation of the shape of the objects was also explored. A component is a region in an object's image which is homogeneous in the hue spectrum. Once the components have been extracted, the relations between these components are represented using non-directed graphs (graph of components). These relations, in general, capture the physical proximity and relative spatial arrangement of components with respect to each other.

Several measures of similarity were presented for comparing these presentations. Three similarity measures were used for feature-based representations - Euclidean similarity (ES), Pyramid match (P_{Δ}) and Manhattan Pyramid Similarity (MPS). P_{Δ} and MPS are multi-resolution measures. In Section 4.2.2, an anomaly was highlighted in pyramid matching by minimization. MPS was proposed to rectify this anomaly. In addition to these similarity metrics, a new metric was also designed to compare the component-based representations.

5 Learning and classification

As previously described in Section 3.3, three visual category learning architectures are explored in this thesis:

1. Instance-based classification with a single classifier;
2. One-class classification; and
3. Instance-based classification with multiple classifiers and meta-learning.

At the implementation level, the key difference between these architectures is the learning and classification approach used. The first and the third architectures use a simple (flat) instance-based approach (see Section 5.1) and/or a cluster-based approach (see Section 5.2) to learn and recognize categories. For these architectures, multiple instance-based and cluster-based classifiers were designed.

In the case of the one-class approach, which is based on the Support Vector Data Description (SVDD) of Tax (2001), a category is described as a hypersphere and represented by the support vectors (boundary instances) of this hypersphere (see Section 5.3.2). In our earlier work, the original implementation of SVDD was modified so as to perform incremental multi-class learning making it suitable for open-ended learning domains (Seabra Lopes and Chauhan, 2007). The work developed in the thesis is built on this previous work which was based on optimization of SVDD parameters using standard quadratic optimization approach, similar to the projection method of (Gill et al., 1981, 1984). The key novelty of the current work is the use of a genetic algorithm for optimizing the SVDD parameters (Section 5.3.3).

Most of the instance-based classifiers are utilized as base classifiers in the multi-classifier architecture. In this approach, further classifiers were developed which combine decisions from those base classifiers (Sections 5.4.2, 5.4.3). This approach also has a significant metacognitive component. One of the meta-level modules, described in Section 5.4.1, maintains success statistics of all the classifiers. The classification decision by the agent is taken by the current most successful classifier. The observed success of the base classifiers is also used by the meta-level component to dynamically reconfigure the classifier combinations.

Instance-based learning approaches are memory intensive, therefore a crucial aspect of the learning agent is the management of it's memory. An approach to memory management that includes selective forgetting is presented in Section 5.5. This approach was implemented and integrated in the multi-classifier architecture.

5.1 Instance-based learning (IBL)

Instance-based learning is the simplest approach followed in this work. Each instance-based classifier can be seen as a combination of a particular instance representation, similarity measure and classification rule.

5.1.1 Category representation

In this approach, each object category is represented by a set of known instances:

$$C_{IBL} = \{\mathbf{y}_1, ..., \mathbf{y}_n\} \tag{5.1}$$

where $\mathbf{y}_i$, $i = 1, ..., n$ are the constituent instances. New instances are stored in the following situations:

– If the user explicitly *teaches* the category of a given object through the *teach* action, the object representation is stored as an instance of the category; if the category was previously unknown, it will be created and initialized with the taught instance.

– After an incorrect agent's prediction of the category of a given object, if the user provides the true category of the object (*corrective feedback*), then the agent adds the object to the set of instances of the category; again if the category is previously unknown, it will be created.

As discussed in Section 3.2, agent's actions are a direct response to user requests, where most of these requests, either implicitly or explicitly, imply classification. The user identifies the need for *corrective feedback* when she notices classification errors in agent's responses.

5.1.2 Classifiers

The instance-based classifiers are designed such that classifying a previously unseen instance involves ranking the known categories according to a measure of membership of that instance to each of the categories. In turn, computing membership measures involves evaluating similarities and/or distances between instances. A category membership measure, M, for an instance-based classifier is a normalization of some object-to-category similarity measure S, according to the following generic formula:

$$M(\mathbf{x}, C_i) = \frac{S(\mathbf{x}, C_i)}{\sum_{k=1}^{N} S(\mathbf{x}, C_k)} \tag{5.2}$$

where $\mathbf{x}$ is a target object, N is the number of categories, C_i is the i-th object category, $i = 1, ..., N$, $S(\mathbf{x}, C)$ is a measure of the similarity between an object $\mathbf{x}$ and a given object category, C. The membership values for the different categories sum to 1.0, allowing their use as evidence in Dempster-Shafer combinations (Section 5.4.2).

These membership measures were used with three classification rules: *average* (AVG) rule, *nearest neighbor* (NN) rule and *nearest cluster* (NC) rule. Classifiers based on the AVG and NN rule were designed for instance-based learning and are presented below. NC classifiers were designed for cluster-based category representations (presented in Section 5.2).

For the average rule, the similarity of a target object $\mathbf{x}$ to a category C is computed by averaging the similarities of that object to the instances of the category:

$$S_{AVG}(\mathbf{x}, C) = \operatorname*{avg}_{\mathbf{y} \in C} S(\mathbf{x}, \mathbf{y}) \tag{5.3}$$

where $S(\mathbf{x}, \mathbf{y})$ is the measure of similarity between target object $\mathbf{x}$ and instance $\mathbf{y}$. Such classification rule is expected to perform better in domains where categories are homogeneous.

In the case of NN classifiers, given an object to be classified, it is compared with all the instances stored in memory. The category containing the instance most similar to the input object is predicted as its category. Thus, in this case, the similarity between the target object $\bar{x}$ and category C is the maximum similarity between $\mathbf{x}$ and the instances in C:

$$S_{NN}(\mathbf{x}, C) = \max_{\mathbf{y} \in C} S(\mathbf{x}, \mathbf{y}) \tag{5.4}$$

The classifiers derived from feature-based representations (Section 4.1.1) use three similarity metrics (*ES*, P_Δ and *MPS*, presented in Section 4.2). For each classification rule described here, the implementation includes 16 base classifiers[1], leading to a total of 32 instance-based classifiers. In the case of the component-based representation approach, a specific similarity

[1](6 feature spaces x 3 similarity measures) = 18 classifiers. For single dimensional features, $AREA$ and $RADSD$, measures ES and MPS equivalent, leading to 18-2 = 16 classifiers per classification rule.

measure was developed (Section 4.2.4). For this special case, only one classifier, based on the NN rule, was developed.

5.2 Cluster-based learning

A cluster-based approach to represent categories is also explored. Since all individual instances are still stored and used for classification of new objects, this approach is still instance-based.

5.2.1 Category representation and learning

Here, given all the instances belonging to a given category, these instances are organized into a set of clusters:

$$C_{CBL} = \{U_1, ..., U_m : U_i = \{\mathbf{y}_{1_i}, ..., \mathbf{y}_{k_i}\}\} \tag{5.5}$$

where each cluster U_i in this set is represented by a set of instances, $\mathbf{y}_{1_i}, ..., y_{k_i}$. The intersection between any two clusters is empty and the union of all clusters gives the set of all known instances of the category.

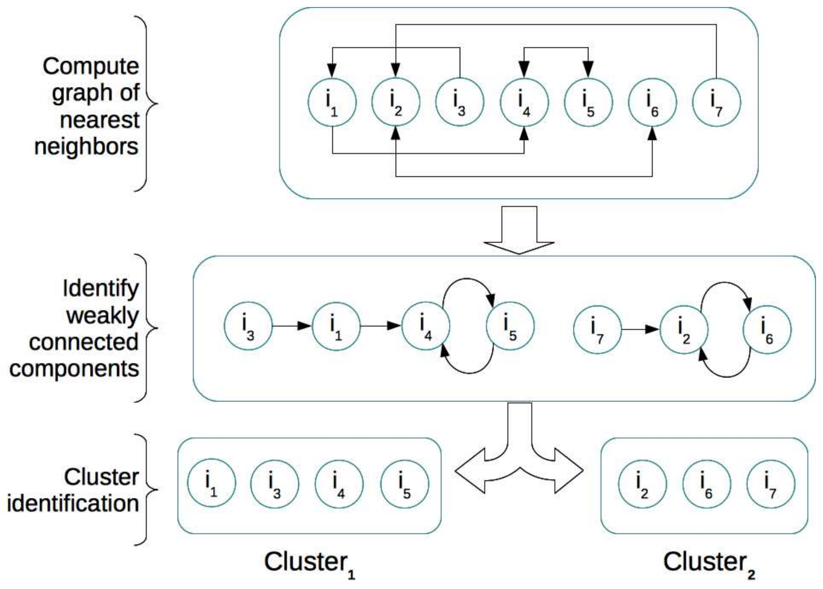
Finding clusters in a given category involves locating the nearest neighbor of each instance in that category. A directed graph is computed, in which edges connect instances to their respective nearest-neighbors. The graph thus obtained may contain one or more weakly connected components (*i.e.* maximal connected subgraphs computed while ignoring edge directions). The set of instances in each of these components forms a different cluster (see Fig. 5.1 for an illustration of the clustering approach).

Similar to the instance-based approach, teaching and correction by the user lead the agent to add a new instance to the taught category. Each time there is a change in the set of instances of a category, the clustering process for this category is run once more, producing a new set of clusters for this category.

5.2.2 Nearest-cluster classification

The instance-based classifiers, based on AVG and NN rules, are complementary to each other in the sense that one exploits the homogeneity (by combining information from all the instances in a category description) and the later makes classification decisions taking advantage of instance specific (local) information. The Nearest-Cluster (NC) rule, was developed to blend these two complementarities.

The classifiers based on the NC rule are applied over the cluster-based category representations described above. For each category, the cluster with the highest average similarity to the target object will provide the membership score of the category. Thus, in this case, the



Compute graph of nearest neighbors
Identify weakly connected components
Cluster identification
Cluster$_1$
Cluster$_2$

Figure 5.1: Extracting clusters from a set of instances representing a category (the nearest neighbor of an instance is pointed by the head of the arrow originating from that instance).

similarity between the target object $\mathbf{x}$ and the category C is the maximum average similarity between $\mathbf{x}$ and the objects in the different clusters $U \in C$:

$$S_{NC}(\mathbf{x}, C) = \max_{U \in C} \left(\operatorname*{avg}_{\mathbf{y} \in U} S(\mathbf{x}, \mathbf{y}) \right) \tag{5.6}$$

The NC rule may have important implications on the robustness and flexibility of the learned model. In the case where a category is composed of a heterogeneous collection of instances (e.g. objects in significantly different poses or clearly different objects from the same category), a specific combination of a feature space and a similarity measure will cluster similar instances together (with respect to that feature space and similarity measure). By bringing the similar instances together, the cluster organization can account for heterogeneous categories, that is, categories which have more than one subcategory associated to them.

Similar to the instance-based classifiers, the NC classifiers are also derived from the feature-based representations, and the corresponding similarity metrics, *ES*, P_Δ and *MPS*.

The NC rule can be used with 16 combinations of feature spaces and similarity measures. In total, combining the three classification rules, similarity measures and feature spaces leads to a total of 48 base classifiers.

5.3 One-class learning and classification

The learning and classification approach used with the one-class classification architecture (Section 3.3.2) is based on Support Vector Data Descriptions (Tax, 2001; Wang et al., 2004).

5.3.1 Support Vector Data Descriptions (SVDD)

In the normal case, SVDD is trained only with positive instances of the target class. Tax (2001) showed that by mapping the data points to a better feature space (by applying a kernel function K on the data), a much more robust and flexible data description can be achieved. Such a description is referred as a hypersphere. Given the set of known instances of a category, the SVDD approach tries to locate the data points (*i.e.* instances) that form a closed description (a hypersphere) around the data. These data points on the boundary of a hypersphere are its support vectors.

Internally, each category is represented by its known instances and the relevant parameters:

$$C_{SVDD} = (\{(\mathbf{y}_1, \alpha_1), ..., (\mathbf{y}_n, \alpha_n)\}, \kappa) \tag{5.7}$$

where $\mathbf{y}_i$, $i = 1, ..., n$, are the stored instances; α_i are the Lagrange multipliers associated with $\mathbf{y}_i$; and κ is the set of parameters associated with the applied kernel function. Intuitively, a Lagrange multiplier, α_i, in the present case, expresses the amount of contribution of an instance y_i in defining the hypersphere boundary (see Section 5.3.2 for further details). The instances for which the Lagrange multiplier is 0 (zero) are considered to lie inside the hypersphere, whereas, the instances with non-zero values are the identified support vectors. In this work, an object is represented using the SSNRA feature space (described in Section 4.1.1).

The criteria for modifying an existing category or for creating a new one are similar to the ones used with the instance-based and cluster-based learning approaches. That is, the *teach* and *correct* actions from the user trigger the learning module.

5.3.2 SVDD learning through Quadratic Optimization

Learning an SVDD from a set of instances implies finding the support vectors. Central to the learning process is an optimization routine which finds the optimal SVDD parameters. The optimization process used to determine the center and the support vectors attempts to minimize two errors:

- Empirical error - percentage of instances of the category that are misclassified.
- Structural error - given by the radius R_h of the hypersphere which must be minimized with respect to the hypersphere center with certain constraints.

In the ideal case (no noise), all training objects can be included in the hypersphere and therefore the empirical error will be 0. In practical applications, however, this may result in over-fitting. Better results can be obtained with not much extra computational expense if a kernel is introduced to get a better data description (Tax, 2001). Tax gives the error L, a combined Empirical and Structural error, to be minimized as:

$$L = \sum_i \alpha_i K(\mathbf{y}_i, \mathbf{y}_i) - \sum_{i,j} \alpha_i \alpha_j K(\mathbf{y}_i, \mathbf{y}_j) \tag{5.8}$$

where $\mathbf{y}_i$ and $\mathbf{y}_j$ are category instances. The following constraints apply on the Lagrange multipliers α_i:

$$\forall i \;\; 0 \leq \alpha_i \leq T; \qquad \sum_i \alpha_i = 1 \tag{5.9}$$

where T gives the trade-off between the volume of the description and the errors. The support vectors with respective $\alpha_i > T$ are considered outside the hypersphere. Tax (2001) states that when SVDD is learned without outliers, T can be set to 1.0 (or larger), indicating that all training data should be accepted. For the experiments conducted in the thesis, T is set to 1.

The kernel K maps the data into a more suitable space, where categories may become more clearly separable than in the original feature space. Although the choice of kernel is data dependent, in most applications the Gaussian kernel produces good results (Tax, 2001):

$$K(\mathbf{y}_i, \mathbf{y}_j) = exp\left(\frac{-\left(D(\mathbf{y}_i, \mathbf{y}_j)\right)^2}{\sigma^2} \right) \tag{5.10}$$

where $\mathbf{y}_i$ and $\mathbf{y}_j$ are the i-th and j-th instances describing a category; σ controls the width of the kernel; and $D(\mathbf{y}_i, \mathbf{y}_j)$ is the Euclidean distance between $\mathbf{y}_i$ and $\mathbf{y}_j$. The Gaussian kernel is also the choice for the system described here. Note that, applying the Gaussian kernel introduces one extra parameter, σ, to the category description.

Using the Gaussian kernel, $K(\mathbf{y}_i, \mathbf{y}_i) = 1$. Thus, ignoring the constants, the minimization problem stated in eqn. 5.8, becomes (with the constraints listed in 5.9):

$$L = -\sum_{i,j} \alpha_i \alpha_j K(\mathbf{y}_i, \mathbf{y}_j) \tag{5.11}$$

Tax (2001) showed that SVDD can also accommodate a few outliers (approximately 5% of training examples) for learning a tighter boundary around the hypersphere. The extension of the error L in eqn. 5.11 to include outliers is trivial. The Langrange multipliers associated with outliers take negative values. Now the new minimization problem, not very different from the eqn. 5.11, becomes:

$$L = -\sum_{i,j} \alpha_i^{'} \alpha_j^{'} K(\mathbf{y}_i, \mathbf{y}_j) \tag{5.12}$$

where $\alpha_i' = l_i \alpha_i$, where l_i are the labels of the instances such that $l_i = 1$ for inliers and $l_i = -1$ for outliers.

Minimization of L with the constraints listed in eqn. 5.9 is a classic quadratic optimization problem, where the problem is to find the optimal values for the Lagrange multipliers α. The original implementation of SVDD used a standard quadratic optimization (`quadprog` function, available with the MATLAB Optimization Toolbox (Branch and Grace, 1996)) of SVDD parameters. `quadprog` implements an active set strategy (also known as a projection method) similar to that of (Gill et al., 1981, 1984) which has been modified for both linear and quadratic programming problems (Branch and Grace, 1996). Our earlier approach to open-ended category learning using SVDD approach used the original implementation of SVDD and category descriptions were learned using largely positive and a few outlier examples (Seabra Lopes and Chauhan, 2007).

To solve this problem, other optimization approaches have been developed (e.g. the active set strategy of Gill et al. (1981, 1984), Sequential Minimal Optimization (SMO) of Platt (1998) and the genetic approach of Tavakkoli et al. (2007)). For the SVDD, the genetic approach of Tavakkoli et al. (2007) led to a more robust and efficient optimization in comparison to other methods.

5.3.3 Genetic SVDD

The standard optimization routine of the original SVDD allows only minimal outlier information to be used to find the optimal support vectors of a class. However, in a multi-class classification scenario, such as ours, as the number of categories learned by the agent increases, there is a much larger number of outliers than the inliers available for any given category. An optimization algorithm which can exploit the extra information provided by the negative instances will potentially be more suitable in a multi-class scenario than the original approach.

As mentioned earlier, the genetic approach for optimizing parameters has been shown to improve the SVDD performance (Tavakkoli et al., 2007). But the approach ofTavakkoli et al. is not incremental. A new optimization approach, using genetic algorithms, is proposed here, which is designed for handling multi-class scenarios. This optimization is included in the normal running of the agent. Ideally, the new optimization approach will lead to minimal or no overlap between different category descriptions.

In the proposed optimization approach, each chromosome contains a possible assignment of values to the Lagrange multipliers of the known instances, and every Lagrange multiplier is a gene in the chromosome. Each time a new instance is added to a category description, a new chromosome is created with as many genes as the number of instances in the description. A new gene is also added to each of the pre-existing chromosomes. The genes of the newly created chromosome are initialized with random values which satisfy the constraints listed in

Algorithm 1 Function to evaluate the fitness of a chromosome.

```
function ChromosomeFitness
returns:
    $f \mapsto$ Fitness of the input chromosome
input:
    $\mathbf{c} \leftarrow$ Input chromosome
    $C \leftarrow$ List of all categories
    $t \leftarrow$ Target category index

$N \leftarrow$ number of categories
$f \leftarrow 0$                                          ▷ Fitness value of $\mathbf{c}$ initialized
$i \leftarrow 1$
repeat

    ▷ Check if $\mathbf{c}$ interferes with the recognition capacity of existing categories
    $\mathbf{y}_i \leftarrow$ randomly chosen instance from $C_i$, $i \neq t$
    if $(NDC(\mathbf{y}_i, C_i) < NDC(\mathbf{y}_i, C_t))$ then          ▷ No interference
        $f \leftarrow f + 0.5$
    else   $f \leftarrow f - 0.5$
    end if

    ▷ Check if $\mathbf{c}$ improves the recognition capacity of instances belonging to $C_t$
    $\mathbf{y}_t \leftarrow$ randomly chosen instance from $C_t$
    if $(NDC(\mathbf{y}_t, C_t) < NDC(\mathbf{y}_t, C_i))$ then          ▷ Correct recognition
        $f \leftarrow f + 0.5$
    else   $f \leftarrow f - 0.5$
    end if

    $i \leftarrow i + 1$
until  $(i > \text{N})$

return $f/(N-1)$
```

eqn. 5.9). The genes of previously existing chromosomes are also modified to be in the range listed in eqn. 5.9. In our implementation, the number of chromosomes is limited to 20. At any moment in time, the genes of the best chromosome for a category description are used as its Lagrange multipliers.

The parameter optimization routine designed for this work is based on finding the most successful chromosome (*i.e.* the best set of Lagrange multipliers) for a given category description. Success of a chromosome **c** for a target category C_t is computed based on the fitness function described in Algorithm 1. The function uses a category membership measure (NDC) which will be presented in Section 5.3.4. Using the genes in **c** as Lagrange multipliers, the fitness of **c** is computed based on the following criteria:

Algorithm 2 Procedure to find the fittest chromosome for a target category.

```
procedure FindFittestChromosome
returns:
    $idx \mapsto$ Index of the fittest chromosome
input:
    $C \leftarrow$ List of all categories
    $t \leftarrow$ Index of target category

$n \leftarrow 0$                                    ▷ Chromosome index initialization
$idx \leftarrow 0$                                  ▷ Fittest chromosome index initialization
$f_{max} \leftarrow 0$                              ▷ Maximum fitness initialization

repeat
    $n \leftarrow n + 1$
    $\mathbf{c} \leftarrow$ n-th chromosome
    $f_n$ = ChromosomeFitness($\mathbf{c}, C, t$)   ▷ Compute fittness of $\mathbf{c}_n$, see Algorithm 1

    if $f_n > f_{max}$ then
        $f_{max} \leftarrow f_n$
        $idx \leftarrow n$
    end if

    if $f_{max} \geq 0.95$ then return $idx$        ▷ Successful chromosome found
until (all chromosomes have been evaluated)

return NULL                                         ▷ None of the chromosomes are suitably fit
```

1. instances from other classes do not get misclassified as belonging to C_t; and

2. stored instances of C_t are correctly classified;

As mentioned earlier, the chromosomes of a given category are modified each time the agent stores a new instance (by adding a new gene that corresponds to the new instance). For this category description, the optimization process attempts to iteratively evolve the set of chromosomes until a chromosome of desired fitness (minimum 0.95) has been found, without affecting the boundary descriptions of other category descriptions. Each iteration involves finding the fittest chromosome. Algorithm 2 lists the steps involved in finding the fittest chromosome.

If no chromosome reaches the desired fitness, a genetic operator is applied to all chromosomes of the target category. Four operators are supported: two crossover operations (heuristic and arithmatic); and two mutation operations (lower boundary and small variation mutations). See the Figure 5.2 for an illustration. An operator is chosen at random for each chromosome. Another SVDD parameter, associated to each chromosome is the width parameter, σ, of the hypersphere. Similar operations are also applied to σ. In the current

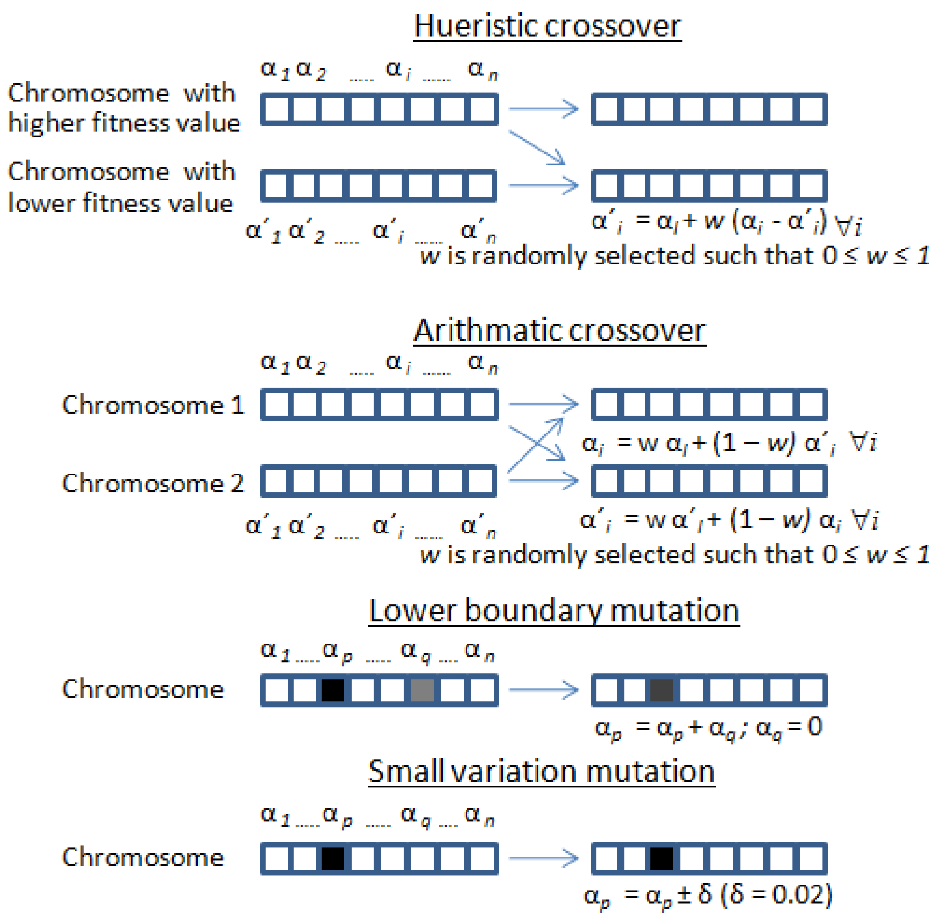

Hueristic crossover
Chromosome with higher fitness value
Chromosome with lower fitness value
$\alpha_1 \alpha_2 \ldots \alpha_i \ldots \alpha_n$
$\alpha'_1 \alpha'_2 \ldots \alpha'_i \ldots \alpha'_n$
$\alpha'_i = \alpha_i + w(\alpha_i - \alpha'_i)\ \forall i$
w is randomly selected such that $0 \le w \le 1$
Arithmatic crossover
Chromosome 1
Chromosome 2
$\alpha_i = w\,\alpha_i + (1-w)\,\alpha'_i\ \forall i$
$\alpha'_i = w\,\alpha'_i + (1-w)\,\alpha_i\ \forall i$
w is randomly selected such that $0 \le w \le 1$
Lower boundary mutation
Chromosome
$\alpha_1 \ldots \alpha_p \ldots \alpha_q \ldots \alpha_n$
$\alpha_p = \alpha_p + \alpha_q;\ \alpha_q = 0$
Small variation mutation
Chromosome
$\alpha_p = \alpha_p \pm \delta\ (\delta = 0.02)$

Figure 5.2: The possible crossover and mutation capabilities of the system (the constraints mentioned in eqn. 5.9 are always maintained). Similar operations are carried out to optimize σ values.

---

implementation, the number of iterations used was 300 (usually the best solution is reached much earlier).

The key advantage of this strategy is that the optimization procedure, instead of minimizing L (eqn. 5.11), tries to find the best set of Lagrange multipliers (and σ) using the classification success of each chromosome while trying to maintain the classification performance for the other existing categories. This makes the optimization process feasible for incremental, online, open-ended and multi-category scenarios.

5.3.4 Category prediction

Once the optimization problem is solved, the membership of a new instance $\mathbf{x}$ to a category C is given as: (Tax, 2001):

$$D(\mathbf{x}, C) = 1 + \sum_{ij} \alpha_i \alpha_j K(\mathbf{y}_i, \mathbf{y}_j) - 2 \sum_{i} \alpha_i K(\mathbf{y}_i, \mathbf{x}) \tag{5.13}$$

$D(\mathbf{x}, C)$ can be interpreted as the squared distance of the input instance $\mathbf{x}$ to the hypersphere center of category C. In a one-class scenario, if this squared distance is less than or equal to the squared radius of a category description, the instance is considered to belong to that category, otherwise $\mathbf{x}$ is considered an outlier. Using the same criterion in a multi-class scenario, more than one class or none of the classes might be identified as the target, and a classification decision will be impossible to make.

For this reason, in our previous work, a more suitable criterion was adopted (Seabra Lopes and Chauhan, 2007). This is also the criterion used in the present work. In particular, a new distance metric called "*Normalized Distance to the Center*" (NDC) was introduced. For a given object $\mathbf{x}$, $NDC(\mathbf{x}, C)$ is the distance of $\mathbf{x}$ to the center of the hypersphere, given as a fraction of its radius:

$$NDC(\mathbf{x}, C) = \frac{D(\mathbf{x}, C)}{R_h(C)^2} \tag{5.14}$$

This normalized distance captures the relative closeness of $\mathbf{x}$ to the center of the category and, therefore, enables comparison of its membership to different categories. Of all the categories that have been learned, the one with the lowest $NDC(\mathbf{x}, C)$ will be considered the most likely category of object $\mathbf{x}$.

Note that, in (Seabra Lopes and Chauhan, 2007), the classification criterion also included a threshold such that if the lowest value of $NDC(\mathbf{x}, C)$ is greater than a threshold (set to 2.0 in (Seabra Lopes and Chauhan, 2007)), the object is considered outside all category descriptions and not belonging to any category. In the work presented here, this threshold is not used, and a target instance is always classified as belonging to the category with lowest NDC.

5.4 Multi-classifier approach with meta-learning (MCML)

Another learning architecture explored in this thesis is based on using multiple classifiers (see Section 3.3.3). As presented in more detail in Section 5.4.1, a meta-cognitive component maintains the success statistics for all the classifiers. These success statistics are updated after each *teach* or *correct* action from the human user. They are used to reconfigure the classifier combinations and to choose the classifier to use for prediction. The final category prediction for a given object is taken from the current most successful classifier. The criteria to store a new instance and to create a new category are the same as in the previous sections.

5.4.1 Self evaluation

In general, on-line self adaptation of intelligent systems relies on self evaluation. In the developed multi-classifier system, the meta-level component of the agent's architecture is responsible for maintaining updated success statistics for all classifiers. Each time the agent sees an object and the user provides its category, the agent runs all classifiers on the object and compares the results with the user-provided category. Then, for each classifier, the respective success measure is updated in a teaching iteration as follows:

$$S_t = w_t S_{t-1} + (1 - w_t) R_t \tag{5.15}$$

where t identifies a teaching iteration (in which the user teaches or corrects the category of a given instance), R_t is the result of the classifier in the t^{th} iteration ($R_t = 1$ if correct category, $R_t = 0$ otherwise) and S_t is the updated measure of success of the classifier in the t^{th} iteration, computed as a weighted average. The weight w_t (which lies in the interval [0, 1]) is the weight of the previous value of the success measure, S_{t-1}. This weight varies in time and is computed with reference to a window of a certain number of iterations, W_t, as follows:

$$w_t = \begin{cases} (t-1)/t, & if \quad t \leq W_t \\ (W_t - 1)/W_t, & otherwise \end{cases} \tag{5.16}$$

While $t \leq W_t$, the success measure equals the arithmetic average of all results R_i so far ($i = 1...t$). For the general case of $t > W_t$, the weight results in gradual forgetting of older results to reflect the most recent performance. The size of the window, also time varying, is decided based on the following rule:

$$W_t = \begin{cases} W_{init}, & if \quad N_t \leq W_{init} \\ N_t, & otherwise \end{cases} \tag{5.17}$$

where N_t is the current number of categories and W_{init} is the initial window size. In the conducted experiments W_{init} was set to 50, that is, initially W_t is 50 (*i.e.* $w_t = 0.98$). Once the number of categories learned by the agent, N_T, becomes greater than 50, the window size is dynamically adjusted to $W_t = N_t$. As the number of categories increase, w_t approaches 1 and older classification results are forgotten relatively slowly[2].

The success measure of a classifier at time t, S_t, is a weighted assimilation of classification results computed over all the iterations. W_t identifies the portion of the history of the classification results with more weight in the current S_t. The classification results from more recent iterations are given more weight while the older ones provide increasingly lower weightage. In general, the contribution of the classification result at time $t - m$ gets discounted by a factor

[2]As an example, for a window of $W_t = 500$ iterations, the corresponding weight $w_t = 0.998$. With these parameters, the result is that the latest 200 iterations (*i.e.* iterations $t - 199$ to t) have a combined weight of approximately 1/3 in the success value, the rest of the window (iterations $t - 499$ to $t - 200$) account for another 1/3 of the success value and all other older iterations (1 to $t - 500$) account for the remaining 1/3 of the success value.

of $\prod_{t-m}^{t-1} w_i$.

5.4.2 Dempster-Shafer combinations

The Dempster-Shafer theory of evidence is a powerful tool for representing and combining uncertain knowledge (Shafer, 1976). It is based on a basic belief assignment, *i.e.* a mass function $m(A)$ that assigns a value in $[0, 1]$ to every subset A of a set of mutually exclusive propositions θ. The belief in the composite proposition $B \subset \theta$ is given by the sum of $m(A)$ for all $A \subset B$. The belief in θ sums to 1.0. In this theory, when multiple evidences allow one to derive multiple basic belief assignments, these evidences can be combined. In particular, two basic belief assignments m_1 and m_2 can be combined by the following rule:

$$m(C) = \frac{\sum\limits_{A,B,A\wedge B=C} m_1(A)m_2(B)}{1 - \sum\limits_{A,B,A\wedge B=\emptyset} m_1(A)m_2(B)} \tag{5.18}$$

This rule is the basis of a well-known method for combining multiple classifiers (Al-Ani and Deriche, 2002; Xu et al., 1992). Each classifier provides evidence that is expressed as a basic probability assignment. In the work of this thesis, the category membership measures (eqn. 5.2) are directly used as masses. As mentioned before, these membership measures are normalized to sum to 1.0 (eqn. 5.2).

Sets containing more than one category are assigned a mass of 0.0, so the approach comes close to the Bayesian combination approach. The main difference is that normalized membership measures are used instead of conditional probabilities. These conditional probabilities could be estimated based on the confusion matrices of each classifier. The classical way of doing this is to acquire a confusion matrix for each classifier in a preliminary training/testing phase. This approach, however, is not viable in a long-term/open-ended learning scenario. In such a scenario, therefore, the alternative would be to build the confusion matrices online. This would imply that, in an initial stage as well as after the introduction of a new category, the conditional probabilities would be heavily biased by the specific cases seen so far. We did some exploratory experiments in this direction and observed that classifier combinations based on conditional probabilities start behaving poorly, but eventually catch up with classifier combinations based on membership measures. However, even in the long run, conditional probabilities did not seem to be able to outperform membership measures significantly, as far as classifier combinations are concerned.

Four Dempster-Shafer classifier combinations were included in the implementation, namely combinations of the top two, three, four and five most successful classifiers (respectively *DS2TOP, DS3TOP, DS4TOP* and *DS5TOP*). As the classification success of each classifier is re-evaluated in each teaching/learning interaction with the human user, these classifier combinations are also dynamically reconfigured in each such opportunity.

5.4.3 Majority voting combinations

Voting methods are also well known in classifier combinations (Kittler et al., 1998; Xu et al., 1992). In the implementation, two dynamically reconfigured classifier combinations based on majority voting were included: majority voting of the top three and five most successful classifiers (respectively *MAJ3TOP* and *MAJ5TOP*). In addition, a classifier combination based on majority voting of all previously described classifiers (*MAJORITY-ALL*) was also included.

5.4.4 Category prediction

The internal computations described up to now culminate in a category prediction that is communicated to the interlocutor(s) of the agent, typically a human user. This category will be the category predicted by the currently most successful classifier, considering all base classifiers and classifier combinations described above.

5.5 Memory management

For instance-based approaches, there is a trade-off between the utility of storing an instance, on one side, and the increase in memory consumption and category membership evaluation costs, on the other. Also, since we focus on long-term learning in an open-ended domain, it must be noted that the instance-based representation of a given category may need to vary in time, as the set of known categories is expanded. In this process, some stored instances, which may have been sufficiently representative at a given point, may become redundant and/or useless (or even misleading) at a later stage. Instances typically become redundant when other instances of the same category are added closer to the category's boundary. A natural way of handling these problems is to develop memory management procedures that result in some form of forgetting.

As long-term learning in artificial agents remains largely an open issue, the same happens with forgetting strategies. Kennedy and Trafton (2007) emphasize that most cognitive systems do not explicitly forget learned knowledge. Forgetting has been addressed for speed-up learning, case-based reasoning and instance-based learning systems. Markovitch and Scott (1988) have shown that random forgetting of up to 90% of learned productions (macro-operators) in a problem-solving domain can improve global performance. Kennedy and Jong (2003) extended the Soar cognitive architecture to remove productions that have not been used for more than some time, achieving statistically better computational performance. Francis and Ram (1993) include case deletion as one of the possibilities for coping with swamping problems in case-based reasoning systems. Mensink and Raaijmakers (1988) implement forgetting by a temporal decrease in the probability of retrieving an item from memory. Montaner et al. (2002) keep a measure of the importance of each memory item, which is strengthened when retrieved and is otherwise decayed in time. Finally, in the incremental instance-based learning approach of Aha et al. (1991), instances are stored only if misclassified and may be eventually deleted. The classification accuracy of individual instances stored in memory is evaluated as new instances are introduced. Based on this information, the stored instances that are believed to be noisy are discarded.

In this work, a single forgetting rule was implemented. The rule for forgetting is basically the same as the rule for remembering. As mentioned, an instance is stored in memory if it cannot be classified correctly. So, in abstract terms, the rule for forgetting is the following:

– An instance can be removed from memory (forgotten) if it will still be classified correctly after removal.

This rule is applied conservatively. Each time the system fails to classify correctly a new instance, leading to addition of the instance to the database, the metacognitive component of the system will check whether any other instance of the same category can be removed. As soon as one instance satisfying the above rule is found, it is removed, and the remaining instances are kept in memory. Only on addition of another instance will the other existing instances be considered for removal. It is to be noted that instance storage/removal decisions are local. That is, these decisions are made without considering the impact of storage/removal of an instance of a particular category on the classification of the instances from other categories.

Although the approach can be applied to the instance-based, SVDD-based and multi-classifier approaches, it was integrated only in the later. In this case, the agent's decision to store or forget an instance is controlled by the performance of the current best classifier. That is, instance storage/removal decisions are biased towards improving the performance of the current best classifier. Category representations get continuously modified to favor correct classification by the current most successful classifier.

5.6 Summary

This chapter focused on the details of the three visual category learning and classification methods, each corresponding to a specific learning architecture presented in Section 3.3.

For the instance-based architectures, learning a category involves simply storing its representative object representations. In the case of IBL, the classifiers are derived from a combination of a specific instance representation, similarity measure and a classification rule. Three classification rules were used: average (AVG), nearest neighbor (NN) and nearest-cluster

(NC). The category membership of new instance then involves evaluating its similarity to the category instances based on the classification rule, and the category with the highest similarity is predicted as the category of that instance.

In the architecture based on one-class learning, a category is described as a hyper-sphere, and represented by its support vectors, the corresponding Lagrange multipliers and the used kernel function. The original SVDD implementation was modified to make it suitable for open-ended multi-class learning scenarios. To utilize the outlier information in the optimization process, a genetic approach was explored for finding the optimal SVDD parameters for each category. Membership of a new instance is given by the category with the smallest NDC (Normalized Distance to the Center).

The multi-classifier architecture (MCML) uses instance-based classifiers as the base classifiers. To combine the classification decisions from the base classifiers, two approaches to classifier combination are explored: Dempster-Shafer combinations and Majority voting. A meta-level component was developed which maintains the success statistics of each classifier, and, at any moment in time, the category prediction is given by the current most successful classifier. The observed success of the base classifiers is also used by the meta-level component to dynamically reconfigure the classifier combinations.

Additionally, a selective forgetting mechanism was also presented with an objective to economize memory usage. Although applicable to all the presented learning and classification approaches, it was implemented only in MCML approach.

6
Grounding spoken words

In standard approaches to grounding vocabulary in artificial agents, words refer to physical objects and object category formation is taken as a supervised learning task that assists in vocabulary grounding (e.g. Chauhan and Seabra Lopes, 2010a; Connell et al., 2012; Gold et al., 2009; Krunic et al., 2009; Seabra Lopes and Chauhan, 2007, 2008; Skočaj et al., 2007; Steels and Kaplan, 2002). This approach makes sense when words are communicated reliably, e.g. as text, and, therefore, there is no ambiguity with respect to the labels of object categories. However, when words are communicated via speech, with all constraints usually associated to speech communication (speaker voice, accent, environment noise, etc.), different utterances of the same word display some amount of variation, leading to possible confusion between utterances of different words. Thus, on the listener's side, interpreting a spoken word involves, firstly, to recognize the word itself, *i.e.* to map the word to a previously known category of spoken words (a word category). The next interpretation step is semantic interpretation, or grounding, understood as the association of a meaning to the heard word.

The conventional approaches to vocabulary grounding either use words directly transmitted in text (e.g. (Cangelosi and Harnad, 2000; Greco et al., 2003); and some of our previous works (Chauhan and Seabra Lopes, 2010a; Seabra Lopes and Chauhan, 2007, 2008)), or, process spoken words using speech recognition tools but ignore the respective recognition uncertainty (e.g. Gold et al., 2009; Krunic et al., 2009; Levinson et al., 2005; Skočaj et al., 2007). There are, however, some notable exceptions. Yu and Ballard (Ballard and Yu, 2003; Yu and Ballard, 2004, 2007) treat a phoneme sequence as a string and use a string matching algorithm (using string changing operations, such as insertion and deletion) to measure the amount of difference between two phoneme sequences. Roy and Pentland (Roy, 2003; Roy and Pentland, 2002) generate a Hidden Markov Model (HMM) from the phoneme sequence predicted for a given speech segment, where each phoneme is assigned an HMM state. The HMM state transitions are strictly left-to-right and the transition probabilities are given by the phoneme models previously trained on a context-independent dataset. To compare two speech segments, they proposed a distance metric which computes the likelihood of producing one speech segment given the HMM of the other speech segment. A similar approach to word representation is taken in this work where the sequence of phonemes, in combination with the Mel-Frequency Cepstral Coefficients (MFCC), extracted from an utterance, represent a spoken word in memory (see Section 6.1.1). For comparing word representations, in addition to the standard “edit distance” metric (Levenshtein, 1966), a new measure, based on dynamic time warping and greedy search, is proposed and presented in Section 6.2.2.

A novel learning and classification approach to support spoken word grounding for artificial cognitive agents is presented in this chapter. The architectural details of this approach were presented in Section 3.4. The architecture for grounding spoken words is an extension of the architecture based on using multiple classifiers, classifier combinations and meta-learning (Section 3.3.3). The latter architecture focused on grounding textually communicated words. This architecture and the underlying learning and classification approach is extended to support spoken word grounding, as described in this chapter.

As in the previous chapters, the work presented in this chapter focused on words that are object category labels. The learning approach is open-ended in that there is no set of words and meanings defined in advance, and new words and meanings are acquired incrementally through interaction with a human instructor. It is assumed that most word categories are reasonably homogeneous when compared to object categories (meanings). This has led us to explore mechanisms that use word categories to dynamically form and reorganize object categories, while word categories themselves are formed and reorganized through clustering of word instances. The strategy to use words to cluster categories is unique and was implemented to test whether, only using the information contained in vocal symbols, visual categories can be learned by a robotic agent.

The rest of the chapter is structured as follows: In the next section, we describe the approach to extract features from a spoken word and present the word and category representation methods. Section 6.2 elaborates the measures of similarity to compare the word representations. Finally, Section 6.3 describes the approach to word category formation and subsequent dynamic formation or update of visual categories.

6.1 Representations

6.1.1 Word representation

Each time the user chooses the option to *teach* or *correct*, the human-robot interaction (HRI) interface waits until the user is ready to speak. Once the user is ready to speak the name, the raw vocal signal is collected in a fixed duration time slot. One of the reasons for setting a limit on the length of the spoken word is that the software (SoX[1]) used for capturing the audio signal from the microphone, requires either an explicit input from the user (Ctl+C *Interrupt* signal from the keyboard) to stop recording, or a time-frame can be defined after which SoX halts the recording. We chose the second method to simplify the interface. After crude testing, the time-frame for word utterances has been set to 2 seconds in the current implementation. This allows a user, unfamiliar with the agent, to conveniently speak a single word within the time limit. A serious drawback of this approach is that, for the same word, different recordings can be out of synchronization. But our hypothesis is that the measures used to compute the similarity between two word representations (see Section 6.2) will be invariant to small unsynchronizations.

From the raw audio signal, two sets of features are extracted: the phoneme sequence; and the Mel-Frequency Cepstrum (MFC) set. MFC provides a good approximation to the response of the human auditory system to a sound stream. This set is obtained using the *wave2feat* tool provided with Sphinx3[2] speech recognition engine. Using this tool with the standard settings, the 2s speech signal is divided into 199 equal sized speech segments. For each of these segments, MFC is calculated, and the 10 most significant initial amplitude values (*i.e.* the first 10 Mel-Frequency Cepstral Coefficients, or MFCC) are stored. In addition, the spoken word, from which the word representation was derived, is also stored in memory.

To extract the phoneme sequence, the *allphone* mode of Sphinx3 is used[3]. This mode predicts:

– the most probable sequence of phonemes for a given speech signal;

– the elements of the MFC set associated with each predicted phoneme.

Once the sound features have been extracted, a word W is represented as:

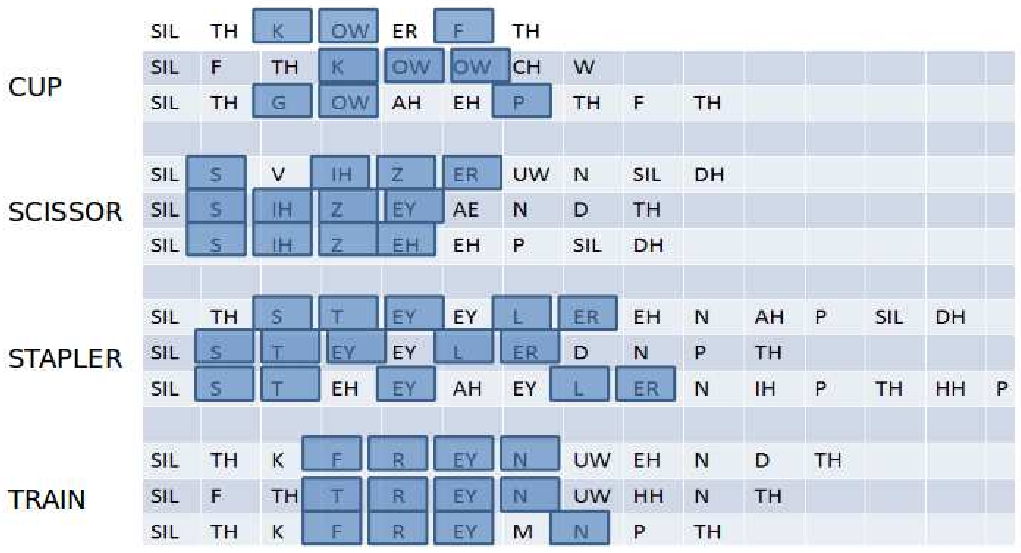
$$W = \{< ph_1, m_1, s_1 >, < ph_2, m_2, s_2 >, .., < ph_n, m_n, s_n >\} \tag{6.1}$$

where n is the number of predicted phonemes, ph_i is the i-th phoneme in the sequence; m_i is the set of all MFCC vectors in the time period for which ph_i was predicted (thus, m_i is a

[1]http://sox.sourceforge.net/

[2]A toolkit for speech recognition based on Hidden-Markov Models (HMM): http://cmusphinx.sourceforge.net

[3]Trained on VoxForge, an open-source speech corpus and acoustic model repository: http://www.voxforge.org



	SIL	TH	**K**	**OW**	ER	**F**	TH								
CUP	SIL	F	TH	**K**	**OW**	**OW**	CH	W							
	SIL	TH	**G**	**OW**	AH	EH	**P**	TH	F	TH					
	SIL	**S**	V	**IH**	**Z**	**ER**	UW	N	SIL	DH					
SCISSOR	SIL	**S**	**IH**	**Z**	**EY**	AE	N	D	TH						
	SIL	**S**	**IH**	**Z**	**EH**	EH	P	SIL	DH						
	SIL	TH	**S**	**T**	**EY**	EY	**L**	**ER**	EH	N	AH	P	SIL	DH	
STAPLER	SIL	**S**	**T**	**EY**	EY	**L**	**ER**	D	N	P	TH				
	SIL	**S**	**T**	EH	**EY**	AH	EY	**L**	**ER**	N	IH	P	TH	HH	P
	SIL	TH	K	**F**	**R**	**EY**	**N**	UW	EH	N	D	TH			
TRAIN	SIL	F	TH	**T**	**R**	**EY**	**N**	UW	HH	N	TH				
	SIL	TH	K	**F**	**R**	**EY**	M	**N**	P	TH					

Figure 6.1: Phoneme sequences predicted in three separate utterances each of the following words: *cup, scissor, stapler* and *train*. The highlighted phonemes for each utterance represent the ideal predictions. The non-highlighted phonemes constitute noise in a predicted sequence.

---

subset of the 199 MFCC vectors initially computed); and s_i is the location of the recorded audio from which the sound features were extracted.

To maintain speaker independence, the Sphinx3 speech recognition engine was not trained on any particular individual or for any specific vocabulary. However, this has a drawback that the predicted phoneme sequence for any spoken word contains a fairly high amount of noise (see Figure 6.1). Therefore, the word representation approach described above uses both the predicted phonemes and the associated MFCC vectors. The underlying assumption is that, in case the phonemes predicted are unreliable, the corresponding MFCC vectors will provide complementary information which can compensate for the incorrect phoneme prediction.

6.1.2 Object representation

The methods presented in Chapter 4 are used here for the visual feature extraction and object representation. Similar to the object representation method used for the multi-classifier with meta-learning approach (Section5.4), an object here is represented using multiple feature spaces. To summarize briefly, different feature spaces capture different aspects of a particular object and are possibly complementary to each other. In total, six feature spaces, previously described in Section 4.1.1, are used:

– 5 feature spaces extracted from the edge-based counterpart of the object image (SLH, SSH, SSNRA, SSNRSD, RADSD). These features spaces are designed to capture the

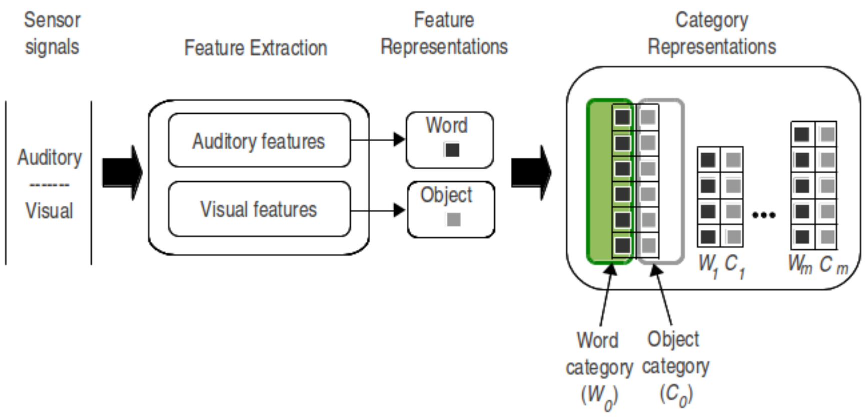


Figure 6.2: Signal perception and representation schema for grounding spoken words.

shape information.

– 1 feature space is composed of a single feature, AREA, which is the only scale-dependent feature.

6.1.3 Category representation

Instance-based representations are adopted both for word categories (spoken words) and object categories. That is, each spoken word category is described by a set of representations of word instances. Likewise, object categories are described by sets of known instances. Each word category and the corresponding object category are coupled together such that each instance in the object category is associated to a word instance in the respective word category. For clarity of understanding, Fig. 6.2 summarizes the signal perception and representation scheme.[4]

6.2 Word similarity measures

To compare word representations two similarity metrics were used. The first metric is the standard *edit distance* (Levenshtein, 1966), defined as the number of edits required to transform one phoneme string to another. Additionally, to take advantage of the information contained in the MFCC set (along with the phoneme string), a new greedy search algorithm was designed to find a locally optimal alignment between two word representations.

[4]This schema is the same as the one shown in Figure 3.5 in Section 3.4, and reproduced here.

6.2.1 Edit distance

The edit distance (also known as Levenshtein distance) is a common string distance metric (Levenshtein, 1966; Navarro, 2001). It counts the number of edit operations (insertion, deletion and substitution) required to transform one string to another. To use this measure, only the partial information from the word representation is used, such that a word W, instead of being represented by eqn 6.1, is represented only by the phoneme string:

$$W = \{ph_1, ph_2, ..., ph_n\}$$

Given the *edit distance*, $ED(W_p, W_q)$, between two words W_p and W_q, their similarity, S_{ed}, is computed by taking the inverse of $ED(W_p, W_q)$:

$$S_{ed}(W_p, W_q) = 1/ED(W_p, W_q) \tag{6.2}$$

Treating phoneme sequence as a string, and using the *edit distance* to compare words, has also been explored in other works (Ballard and Yu, 2003; Yu and Ballard, 2004, 2007). However, as is evident, this metric does not use the information contained in the MFCC set. We assume that the MFCC set (in addition to the phoneme sequence), could provide a more robust comparison between word representations. In the following sub-section, we will discuss a novel similarity measure designed with this specific aim.

6.2.2 Phoneme-MFCC similarity measure

A novel similarity measure was developed that took advantage of the combined information contained in the phoneme sequence and its associated MFCC set (eqn. 6.1). Given two word representations[5]

$$\begin{aligned} W_p &= \{< ph_{p,1}, m_{p,1} >, < ph_{p,2}, m_{p,2} >, .., < ph_{p,r}, m_{p,r} >\} \\ W_q &= \{< ph_{q,1}, m_{q,1} >, < ph_{q,2}, m_{q,2} >, .., < ph_{q,s}, m_{q,s} >\} \end{aligned}$$

an algorithm based on dynamic time warping (DTW) (Rath and Manmatha, 2003) is used to find the distances between each $m_{p,i}$ and $m_{q,j}$, where i=1,2,...,r, j=1,2,...,s, and r and s are the number of phonemes in the phoneme sequence of W_p and W_q respectively. The end product of this algorithm is an $r \times s$ matrix where each element (i,j) of the matrix is the DTW distance measure for m_{pi} and m_{qj}. We will refer to this matrix as $DTW(W_p, W_q)$.

In an ideal condition, for two words in the same category, the sequences of phonemes (and respective MFCC sets) should be exactly the same. This is exemplified in the case of $DTW(W, W)$, where each diagonal element will be zero. The diagonal path provides an approximation of the match between the sequences of phonemes from two different words.

[5]The recorded speech corresponding to a word representations is not used for computing this similarity. Therefore these sounds are not shown as the part of the word representations here.

The summation of diagonal values of the DTW matrix is one possible distance measure for comparing words. However, in real time applications, such measure will lead to a very weak performance because of the noise present in the predicted phoneme sequence (Figure 6.1). Alternatively, a local greedy search algorithm has been designed to find a locally optimal path in the DTW matrix such that the sum of all the elements leads to a locally optimal minimum. The objective is to reduce the total cost while maintaining proximity to the diagonal path.

Given two words W_p and W_q, this cost is given by:

$$C(W_p, W_q) = \sum_{i=1}^{min(r,s)} c_i(W_p, W_q) \tag{6.3}$$

where

$$c_i(W_p, W_q) = \begin{cases} 0, \quad if \quad ph_{p,i} \in \{ph_{q,i-1}, ph_{q,i}, ph_{q,i+1}\} \\ \\ min(\quad DTW_{i,i-1}(W_p, W_q), \\ \qquad DTW_{i,i}(W_p, W_q), \quad DTW_{i,i+1}(W_p, W_q) \quad), \quad otherwise \end{cases}$$

is the cost function, which returns the distance between a diagonal element in W_p and its closest neighbor in a local search window around the corresponding element in W_q. In the case of an exact phoneme match within the search window, c_i is zero. Otherwise, c_i would be computed based on the DTW distances. It is essential to note that the cost function $C(W_p, W_q)$ is not symmetric and thus not a true distance measure. Thus, the final similarity measure between two words is calculated based on the minimum cost as follows:

$$S_{dtw}(W_p, W_q) = \frac{1}{min(C(W_p, W_q), C(W_q, W_p))} \tag{6.4}$$

6.3 Learning and classification

This section introduces a novel strategy, which has been designed to facilitate object category formation by taking the association between names and their meanings into account. This strategy uses the information contained in the names (word categories) for dynamic formation of meanings (object categories).

6.3.1 Dynamic category formation and update

Although the objects belonging to a certain category can be very different from each other and different object categories can share a single name, different instances in a word category will often be relatively similar to each other. In this work, we assume that most word

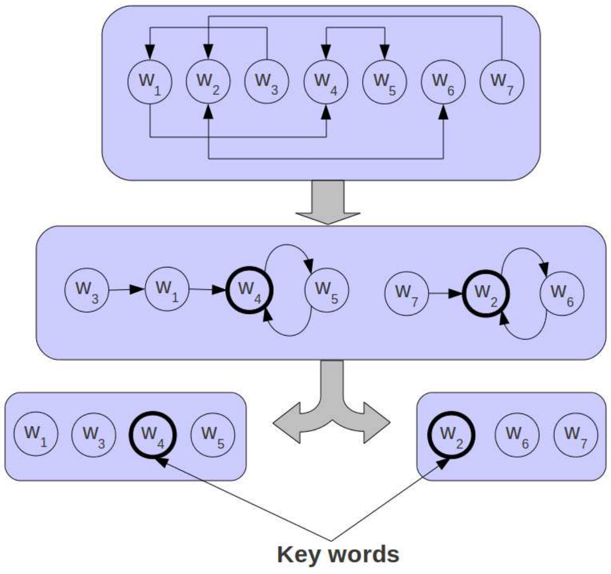


Figure 6.3: Extracting clusters from a set of seven words representing a category name (the nearest neighbor of an instance is pointed by the head of the arrow originating from that instance); key words are the word instances with the highest number of nearest neighbor links in a given cluster.

---

categories are reasonably homogeneous and therefore should not contain more than one easily recognizable cluster of instances. Given this assumption, the presence of two or more clusters suggests that the word category actually contains instances of two or more words. Based on this, a novel methodology has been designed that uses word clustering to dynamically form/organize not only word categories, but also object categories.

The clustering process begins with a word classification routine. Each time there is a *teach* or a *correct* action performed by the human user, the user utters the category name of a selected object. A classifier based on the nearest-neighbor (NN) rule is used for classifying this input word. Here, a "word" to be classified is compared with all the word representations stored in memory. The word category, W_i, containing the instance most similar to the input word is predicted as the category of that word. The input word instance (the name) is then added to W_i, and the corresponding object instance is added to O_i, the object category coupled with W_i.

The addition of a new word to a word category initiates the clustering process for that category. The cluster identification process involves locating the the nearest neighbor, using a similarity metric which can can either be S_{ed} (eqn. 6.2) or S_{dtw} (eqn. 6.4) of each instance. The word category is then represented as a directed graph, in which edges connect instances to their respective nearest neighbors. One or more weakly connected components (*i.e.* maximal connected subgraphs computed while ignoring edge directions) will be identified in the graph.

Algorithm 3 Dynamic category formation and organization(W, O, i)

```
$W$ - array of all word categories (input/output)
$O$ - array of all object categories (input/output)
$i$ - index of a particular category in W and O (input)

$Clusters \leftarrow$ clusters formed for $W_i$
$p \leftarrow$ number of clusters in $W_i$
if $p = 1$ then return

repeat
    $w_{key} \leftarrow$ key word for $Clusters[p]$
    $s \leftarrow$ max_similarity($w_{key}$, $Clusters[p] - w_{key}$)    ▷ the measure of similarity between $w_{key}$
                                                                       and its closest neighbor
    $c \leftarrow 0$
    $m \leftarrow i$
    repeat
        if $c \neq i$ then
            $s_{max} \leftarrow$ max_similarity($w_{key}, W_c$)
            if $s_{max} > s$ then
                $s \leftarrow s_{max}$
                $m \leftarrow c$
            end if
        end if
        $c \leftarrow c + 1$
    until c = number of known categories
    if $m \neq i$ then
        Move the word instances in $Clusters[p]$ to $W_m$ and the respective object instances to $O_m$
    else
        Create a new word category for $Clusters[p]$ and form a new object
        category description from the respective object instances
    end if
    $p \leftarrow p - 1$
until $p = 0$
Remove $W_i$ and $O_i$
```

The instances in each of these components will form a separate cluster (see Figure 6.3). For each cluster, the instance with more edges pointing to it in the graph is defined as its "key word", or w_{key}.

In the case where two or more clusters are identified for a given word category, W_i, each of them is checked if it should form a completely new word category, or if it should be merged to any of the other existing word categories. A given cluster in W_i will form a completely new word category if all instances from all other word categories are less similar to the keyword of that cluster, W_{key}, than the nearest neighbor of W_{key} in that cluster. Otherwise, the cluster will be merged to the category W_m, $m \neq i$, containing the closest instance to W_{key}. At the same time, the associated object descriptions are moved to the corresponding object category. The complete procedure is described in Algorithm 3.

6.3.2 Object classification

The user can perform an *ask* action by selecting an object in the scene and requesting its category name. This action triggers the agent's classification routine which is based on the multi-classifier system approach, previously discussed in Section 5.4. Two types of classifiers are included in the implementation:

– 16 base classifiers based on NN and NC rules[6] (Section 5.1); and

– 7 classifier combinations, based on majority voting and Dempster-Shafer evidence theory (Sections 5.4.2 and 5.4.3).

A meta-learning component maintains the updated success statistics for all the classifiers and, based on these statistics, reconfigures classifier combinations (also discussed in Section 5.4.1). The final category prediction result for a given object is taken from the current most successful classifier, and the audio associated with the key word of this category is reproduced as the response to the user.

6.4 Summary

This chapter presented an approach for grounding spoken words. This approach is an extension of the architectures designed to ground unambiguous object names (such as those typed directly into the computer terminal). Unlike text, communication using speech introduces errors and unreliability related to spoken communication.

The first computational step in grounding a word involves recognizing/classifying the word itself. In our case, a *word instance* is represented by a set of phonemes and the 10 most significant mel-frequency cepstral coefficients associated with each phoneme, and a simple instance-based approach is used for representing the *word categories*. To compare word instances, two measures were presented: Edit distance (Levenshtein, 1966) and a similarity measure based on dynamic time warping and greedy search. Classifying a spoken word is based on the nearest-neighbor rule.

Similarly, the instance-based approach is also used for representing the object categories, and the object classification approach is based on using multiple classifiers and classifier combinations. Finally, a word category is grounded in its object category by coupling instances in the word category with instances in the corresponding object category.

A new method was presented in Section 6.3.1 which uses the word categories to dynamically form and reorganize object categories. The approach is based on applying clustering to the set of word instances in a word category, leading to either transfer of clusters between

[6]This work was carried out during 2009-2010, when only a subset of the classifiers, presented in this thesis, were implemented. Additionally, the similarity measure MPS was also not developed. This is one of the reasons for having only 16 base classifiers in this implementation.

categories or creation of new categories. For each *teach* or *correct* action from the human user, this clustering routine is called and relevant word and object category descriptions are modified or updated.

7 Classical evaluation

The instance-based and cluster-based classifiers described in Chapter 5 were designed to take advantage of the relation between specific (features-based) instance representations, similarity measures and classification rules. The performance of these classifiers is a direct reflection of the quality/suitability of such combination of ingredients. Due to practical reasons related to how the developed system and this thesis document evolved, we focus this chapter on the classifiers used as base classifiers in the multi-classifier architecture. These classifiers are evaluated on several datasets using classical cross-validation methods. This type of evaluation provides a straightforward base for comparing the different classifiers among themselves and possibly with other approaches described in the literature.

Moreover, in the multi-classifier architecture, evaluation of the externally observable performance does not provide any assessment of the base classifiers. Although, the performance of base classifiers and classifier combinations can be evaluated based on the internally computed success measures (Section 5.4.1), there is a certain interdependency among the classifiers in the integrated system. In fact, when a new instance is considered for storage, what is taken into account is the global performance of the agent, which, in turn, reflects the performance of the best classifier (including combinations) selected for classifying the instance. Thus, each classifier individually may have limited influence on which instances are stored, and, by consequence, may have to classify based on stored instances that may be misleading for that particular classifier.

Word learning is inherently a multi-class problem. Therefore the performance evaluation metrics used for assessing classification approaches should be suitable for multi-class scenario.

Most of the literature on classifier evaluation metrics is geared towards binary (two-class) classification scenario and it is generally assumed that these metrics can be directly extended to multi-class scenarios (Holt et al., 2010; Sokolova and Lapalme, 2009). In the next section, we will discuss various available metrics for evaluating multi-class classifiers, leading to the final selection of metrics that are most suitable for the classification approaches proposed in this thesis.

All classifiers were designed for open-ended scenarios, thus allowing anytime addition of not only new instances (*i.e.* instance representations) but also new categories. That is, new categories are created and existing categories are modified continuously as new instances are introduced to the agent. In our case, addition of new instances to the agent is non-trivial. Instances are added to the category descriptions or new categories are created only if the human instructor specifically performs a "*teach*" or "*correct*" action. The assumption is that only a few essential instances are sufficient for category generalization. This approach differs from non-incremental instance-based learning algorithms (also known as, one-step learning algorithms), where a category is represented by all the known instances belonging to that category. Although the one-step approach leads to category descriptions with more information, it has greater memory requirements, classifier predictions are computationally more expensive and many of the stored instances are redundant.

Given that the incremental learning approach comes closer to our open-ended learning scenario, the performance of the base classifiers is evaluated not only with one-step, but also with incremental learning approaches at the task of instance-based object classification. The objective is to estimate and compare, for each base classifier, the generalization capabilities of the learned category models using the one-step and incremental approaches. The evaluation methodology is described in Section 7.3.

These evaluations are conducted using several datasets described in Section 7.2. The results obtained over one-step and incremental evaluations are reported and discussed in Sections 7.4 and 7.5 respectively. The results are further analyzed to choose an appropriate set of base classifiers for the multi-classifier apparoach (Section 7.6). Classifier configurations for word classification, presented in Chapter 6, are evaluated in Section 7.7.

7.1 Evaluation metrics for multi-class classification

In machine learning literature, most of the metrics for evaluating supervised learning algorithms have been designed for binary classification scenarios where each object can be classified as positive or negative (Fig. 7.1a). Typically, evaluation of a binary classifier is based on the number of correct/incorrect predictions it makes on a given test dataset. Given a binary classifier, a convenient way to look at its classification results over a test set is a confusion matrix (also known as the contingency table)(Table 7.1, see also the illustration in Figure 7.1b). Some of the standard evaluation metrics in machine learning literature are *accuracy, precision* and *recall.* These metrics can be extracted directly from the confusion matrix and are given as:

– *Accuracy*: Fraction of correct classifications.

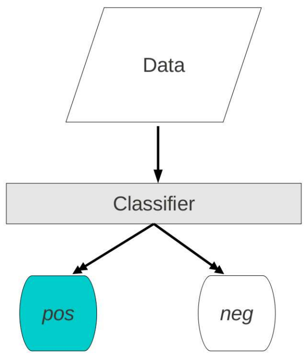
$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{7.1}$$

– *Precision*: Ratio of "number of correct predictions for *pos* class" to "total number of instances predicted as *pos*".

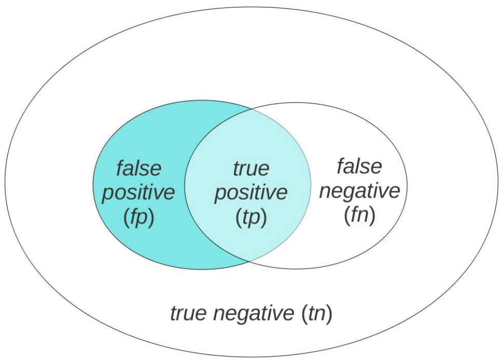
$$Precision = \frac{tp}{tp + fp} \tag{7.2}$$

– *Recall*: Ratio of "number of correct predictions for *pos* class" to "total number of instances of *pos*".

$$Recall = \frac{tp}{tp + fn} \tag{7.3}$$



(a) Binary classification



(b) Binary classification evaluation variables

Figure 7.1: Binary classification

		Predicted class	
		pos	neg
True class	pos	true positive (tp)	false negative (fn)
	neg	false positive (fp)	true negative (tn)

Table 7.1: Confusion matrix for binary classification

Each of these metrics identifies different aspects of classification decisions, and their suitability (and limitations) at evaluating a classifier depends on the classification task at hand

(Manning et al., 2008, pg. 155-156). However, depending on the classification task, these metrics may not be sufficient to assess the quality of a classifier. Therefore many other single-value metrics (derived from the above stated metrics) are also common in machine learning literature - for example, *LIFT, F_measure, kappa statistic*, amongst others (Manning et al., 2008).

Keeping within the context of this thesis, here, we will not delve into details of suitability and limitations of each of these (and many other) metrics, which were designed for the binary classification scenario. The focus here is on evaluating the performance of multi-class classifiers and the choice of evaluation metric will be based on its suitability.

A classifier that is designed to deal with more than two classes is a multi-class classifier. Multi-class classification is generally divided into two broad categories (Qi and Davison, 2009): single-label classification, where each test sample is assigned to one and only one class; and multi-label classification, where a test sample can be assigned to one or more classes. In our case, the classifiers were designed to perform single-label classification. To evaluate multi-class, single-label classifiers, multiple metrics exist that are generalizations (with modifications to suit multi-class scenario) of the standard metrics for binary-classification described above. For an N-class classification problem, classifier evaluation involves testing on independently drawn and previously unseen instances from these classes. Each test instance gets assigned to one of the N classes leading to an $N \times N$ confusion matrix, A.

Table 7.2 lists some of the key metrics used for evaluating multi-class single-label classification (taken from the review in Sokolova and Lapalme (2009), except *Accuracy* which was taken from Holt et al. (2010)):

To choose an appropriate evaluation metric, a further division of the multi-class, single-label classification problem is made based on the classification approach:

1. Type-a: Given an instance to be classified, the classifier predicts it as belonging to one of the known classes or to an unknown class. This is usually the case when the classification approach is based on thresholding. Here, if the class-membership decision (using, for example, similarity, probability) lies below a certain threshold, the instance will be identified as belonging to an unknown class; and

2. Type-b: Given an instance to be classified, the classifier always predicts it as belonging to one of the known classes.

Classifiers explored in this thesis were designed to follow the Type-b approach. In other words, these classifiers do not support the classification of a given instance as unknown. Given an instance to be classified, the base classifiers will always predict it as belonging to one of several possible classes. Therefore, an incorrect class prediction of an instance is both a false positive (for the predicted class) and a false negative (for the true class). This has important implications on the multi-class evaluation metrics, because in this scenario,

Metric	Formula
Accuracy	$\frac{1}{\lvert A\rvert}\sum_{i=1}^{N} A_{i,i}$
Average accuracy	$\frac{1}{N}\sum_{i=1}^{N}\frac{tp_i+tn_i}{tp_i+tn_i+fp_i+fn_i}$
$Precision_\mu$	$\frac{\sum_{i=1}^{N} tp_i}{\sum_{i=1}^{N}(tp_i+fp_i)}$
$Recall_\mu$	$\frac{\sum_{i=1}^{N} tp_i}{\sum_{i=1}^{N}(tp_i+fn_i)}$
$Precision_M$	$\frac{1}{N}\sum_{i=1}^{N}\frac{tp_i}{tp_i+fp_i}$
$Recall_M$	$\frac{1}{N}\sum_{i=1}^{N}\frac{tp_i}{tp_i+fn_i}$

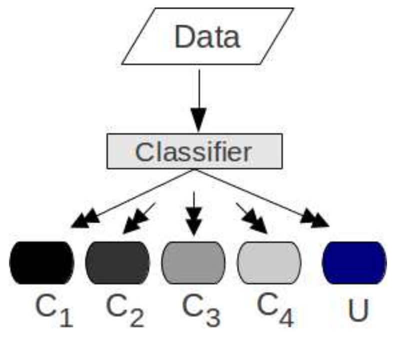
Table 7.2: Evaluation metrics for multi-class single-label classification where $|A| = \sum_{i,j} A_{i,j}$ is the total number of predictions, tp_i are true positives, fp_i are false positives, fn_i are false negatives and tn_i are true negatives, all for class i; indices μ and M represent micro and macro averaging. The difference between micro and macro averaging is that macro-averaging gives equal weight to each class, whereas micro-averaging gives equal weight to each prediction (this also implies that micro-averaging will favor bigger classes)(Sokolova and Lapalme, 2009).

Accuracy, $Precision_\mu$ and $Recall_\mu$ will lead to the same result. Note that this implication is not valid for the Type-a classification approaches.

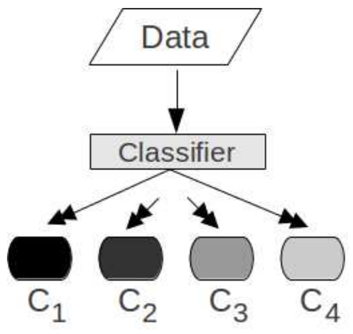
The evaluation of classifiers, in our case, involves giving equal weight to each instance prediction. Therefore, macro-averaging measures will not be considered. *Average accuracy* is also not a suitable evaluation measure for the present scenario, because it overestimates the the accuracy of a multi-class learning algorithm (see Appendix B for a detailed analysis).

To summarize, the primary evaluation measure will be:

$$\boxed{Accuracy = \frac{1}{|A|}\sum_{i=1}^{N} A_{i,i} \qquad (7.4)}$$



(a) Multi-class classification into known classes (C_i) or as unknown (U)



(b) Multi-class classification into known classes only

Figure 7.2: Two scenarios of multi-class, single-label classification

Another metric, specifically for evaluating the memory usage, is the "number of instances stored per category". This metric, along with *Accuracy*, will assist in comparing the models learned using one-step and incremental learning approaches.

7.2 Datasets

The experimental evaluation of different classification methods, reported in this chapter, was carried out on four object images datasets: LANGG68 (Seabra Lopes and Chauhan, 2008) which contains 7350 color images of 68 objects; COIL-100 (Nene et al., 1996) which contains 7200 color images of 100 objects; ALOI-1000 (Geusebroek et al., 2005), the largest dataset, containg 110,250 images of 1000 objects; and ETH80 (Leibe and Schiele, 2003) consisting of 3526 images of objects from 8 object categories. For all datasets, the object images were captured using a fixed camera against a simple background and each image contains a single object.

LANGG68 was collected by us during the period of 2007-2008. To avoid obtaining a biased (perhaps, overly optimistic) view of the performance, experiments were also carried out on publicly available datasets. The selected datasets (COIL-100, ETH80 and ALOI-1000) are highly popular in the computer vision literature for evaluating visual classification methods.

Some of the main differences between these datasets can be highlighted before going into specific details of each of them. One key difference between these datasets is that the categories in ETH80 contains images of more than one object, whereas each category in LANGG68, COIL-100 and ALOI-1000 strictly contain images of the same object. In LANGG68, all images of an object were captured with the same view, whereas for COIL-100, ALOI-1000 and ETH80, the images of an object contain multiple views. The COIL-100 dataset is the only evenly distributed dataset, that is, each category has exactly 72 images. Images in

ALOI-1000 are clearly more noisy (caused mainly by the changes in illumination) than those in the other datasets. The following subsections describe these datasets in more detail.

7.2.1 LANGG68

The LANGG68 dataset was gathered by us in the framework of a project funded by FCT (Portuguese Science Foundation): LANGG - Language Grounding for Human-Robot Interaction. From many experiments involving interactions between a human instructor and the agent, a dataset of 7350 images of real world objects (from 68 categories) has been collected (examples of such experiments can be found in Seabra Lopes and Chauhan (2007, 2008)). A human user, in the normal execution of our agent, showed different objects and provided corresponding names to the agent. Images of the objects shown during these experiments were captured and stored to create this dataset. Each image in the dataset contains a single object in black background. Different views of the same object were handled as different categories. All images were taken with a low cost firewire camera placed in a fixed position above the table.

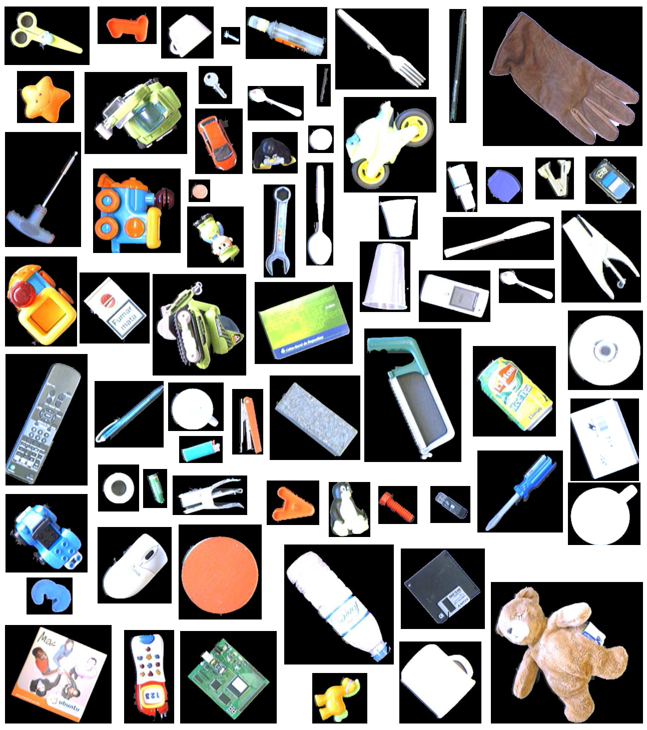


Figure 7.3: Sample images of objects in LANGG68 dataset

These 68 categories can be roughly grouped as follows: 40% are office objects; 20% are child toys; 20% are other home objects; and the remaining 20% are objects of varied types. Figure 7.3 displays one sample image per category and can give an idea of the type of images in this dataset. These object images (and their names) are a direct result of how the human showed the objects

7.2.2 COIL-100

COIL-100[1] (Nene et al., 1996) is a publicly available dataset of 7200 color images of 100 different objects (with 72 images/object, where each image corresponds to a different pose of that object).

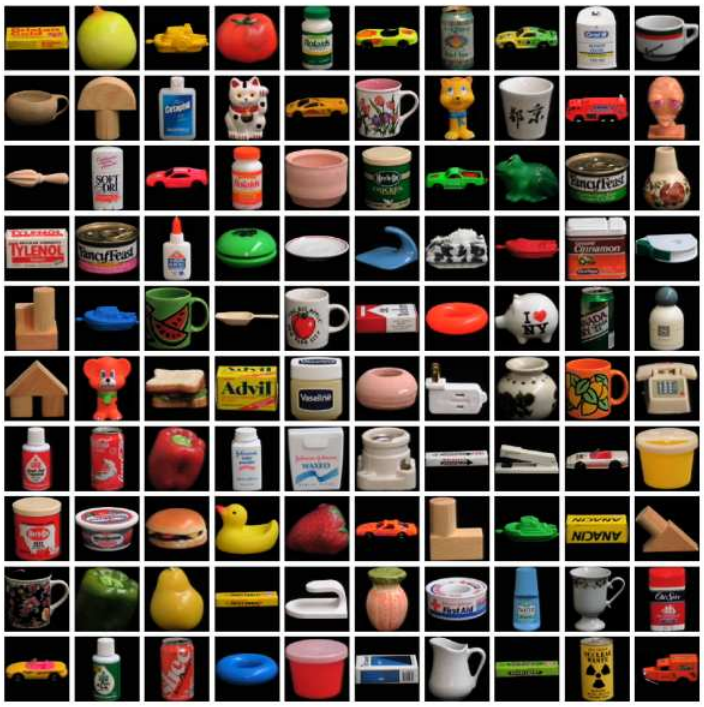


Figure 7.4: Sample images of objects in COIL-100 dataset (Nene et al., 1996)

The object images were captured from a fixed CCD color camera and the objects were placed on a turn-table against a black background. From each captured image, a rectangular bounding box, containing the object, was extracted as a separate image. These bounding box images were then resized to 128x128 (while preserving the aspect ratio) and further normalized with respect to the intensity values. It is these normalized images that form the COIL-100 dataset (see Figure 7.4 for one sample image per object in this dataset).

[1]Columbia University Image Library (COIL-100) dataset is available at:
http://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php

7.2.3 ALOI-1000

ALOI-1000[2] (Geusebroek et al., 2005) is another publicly available dataset which contains 110250 images of 1000 different objects[3]. Amongst all the datasets used for evaluation, this is the largest, and the images in this dataset are relatively more noisy than in the other three datasets. Overall, this dataset contains slightly above 100 images per object.

Similar to COIL-100 dataset, the object images were captured from a fixed CCD color camera and the objects were placed on a turn-table against a black background. These images were captured in different sensory variations where object pose (72 different orientations), illumination angle (24 different angles) and illumination color (12 configurations for illumination color temperature) were systematically changed. Additionally, only from 750 objects, wide baseline stereo images were captured where two cameras were used to gather the stereo information. A stereo image is then stored as 3 different flat images (stereo information can be derived from combining these images). Since the feature extraction and representation approaches developed in this thesis were designed for 2D images, the stereo information is not exploited in the experiments (that is, only the flat images are used in our experiments).

7.2.4 ETH80 dataset

ETH80[4] is a publicly available dataset which consists of 3526 images of natural as well as human-made objects from 8 different categories (Leibe and Schiele, 2003).

This dataset differs from the previous three in the sense that each category is composed of 10 to 13 different objects that belong to that category. For each object, 41 color images are captured from differing viewpoints. These images were captured using Sony DFW-X700 progressive scan digital camera with 1024x768 pixel resolution and the objects were placed on a table against a blue chroma keying background. Figure 7.5 shows one sample image per object in the ETH80 dataset.

For each image, the dataset includes a segmentation mask (see Figure 7.6a,b). Using this mask, the original image can be easily extracted from the background (see Figure 7.6c). By applying the corresponding mask to each image in the dataset, a new collection of images was created which only included the object with black background (similar to the images in LANGG68 dataset). All the reported experiments on ETH80 were carried out on this new set of images.

[2]Amsterdam Library of Object Images (ALOI-1000) dataset is available at:
http://staff.science.uva.nl/ aloi/

[3]See ALOI-1000 sample images at:
http://staff.science.uva.nl/ aloi/www-images/overview.html

[4]ETH80 dataset is available at:
http://www.d2.mpi-inf.mpg.de/Datasets/ETH80

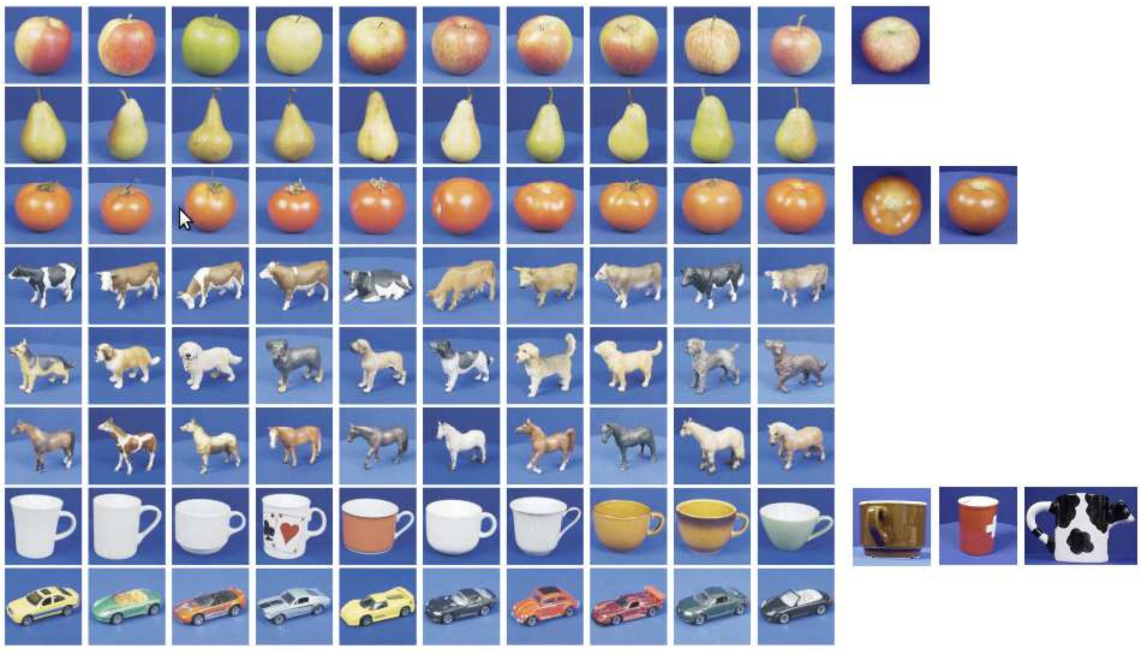


Figure 7.5: Sample images of objects in ETH80 dataset. First 10 columns show sample images of the original objects from Leibe and Schiele (2003) and the last 3 columns show sample images of the recently added objects by the authors of the dataset.

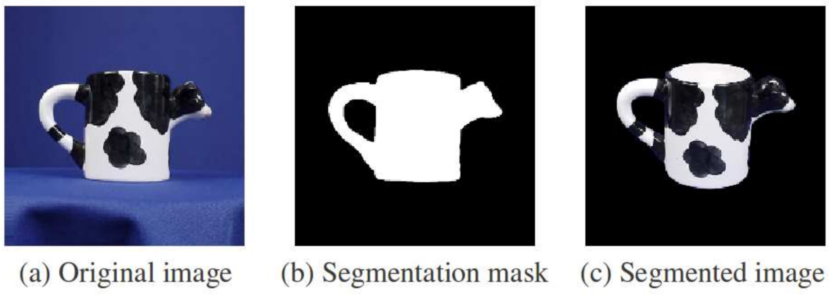


Figure 7.6: (a) An example image from the category *cup* in ETH80 dataset; (b) its corresponding mask; and (c) the object image extracted from the original image after applying the mask.

7.3 Methodology

To assess the performance of each classifier, the k-fold cross-validation procedure was used. This is one of the most widely used methods for estimating the generalization performance of a learning algorithm as well as for comparing the performances of two or more learning algorithms (Refaeilzadeh et al., 2009).

The folds are created using stratified sampling (a preferred configuration when using

k-fold cross-validation (Kohavi, 1995; Refaeilzadeh et al., 2009). In this case, given a multi-class dataset, stratified k-fold cross-validation involves dividing the dataset into k equal sized subsets, where each subset contains examples from all the classes, with approaximately $1/k$ of all the examples from each class. None of the examples are used more than once when creating these subsets, *i.e.* the subsets are mutually exclusive. In particular, for k-fold cross-validation, setting k to 10 is a commonly used and a generally recommended configuration (Kohavi, 1995; Refaeilzadeh et al., 2009).

Algorithm 4 Incremental training procedure

```
procedure Incremental_Training($S$)
returns: Set of category models
input:
    $S \leftarrow$ Randomly organized set of all training instances

$C \leftarrow$ Set of empty category models for all categories in $S$
$n \leftarrow$ number of instances in $S$

for $i$ in $1...n$ do
    $trueCat \leftarrow$ the true category of $S_i$
    $predCat \leftarrow$ Classify $S_i$

    if $predCat \neq trueCat$ then
        update $C_{trueCat}$ with $S_i$
    end if
end for

return $C$
```

Each base classifier configuration includes the choice of a feature space (*AREA, RADSD, SLH, SSH, SSNRA* and *SSNRSD*), similarity measure (*ES*, P_{Δ} and *MPS*) and the classification rule (*AVG, NN, NC*). Note that for the single-dimension feature spaces, *AREA* and *RADSD*, the Manhattan-Pyramid similarity measure, *MPS*, is equivalent to the Euclidean similarity, *ES*. This leads to a total of 48 different configurations.

For each configuration, two 10-fold cross-validation experiments were carried out on every dataset:

1. In one experiment, all the instances in the training set are used to describe object categories (one-step learning scenario).

2. In the other experiment, instances from the training set are introduced to the agent incrementally in a random sequence (incremental learning scenario). Each introduced instance is classified based on the previously stored knowledge (training of the agent begins with no initial knowledge). In case of a misclassification, the instance representation is added to the set of instances belonging to the correct category (this is akin

Table 7.3: Average performance of the different shape signatures, similarity measures and decision rules using one-step learning.

	Average acc.(%)			
	LANGG68	COIL-100	ETH80	ALOI-1000
Similarity measure				
ES	62.6	52.9	53.8	45.1
MPS	**65.4**	**56.4**	**57.0**	**47.4**
$P_\triangle$	61.9	51.5	54.3	41.8
Feature space				
SLH	61.9	53.4	57.3	38.8
SSH	59.8	**58.2**	55.8	**50.6**
SSNRA	**73.5**	55.9	**57.9**	49.5
SSNRSD	57.9	46.8	49.1	40.2
Decision rule				
AVG	50.3	29.1	41.5	20.6
NC	65.8	60.0	57.9	50.9
NN	**73.7**	**71.7**	**65.6**	**62.8**

to *corrective feedback*). Otherwise, that particular instance is ignored. Algorithm 4 describes the training procedure.

7.4 Results obtained with one-step learning

For classifiers trained using one-step learning, the different base classifier configurations and respective results are listed in Table C.1. These results are summarized in Table 7.3 which shows the average accuracies obtained with different shape signatures, similarity measures and decision rules (classifier configurations involving single-dimensional features spaces are shown in Table 7.4).

With regards to the type of classifier, the top performing classifier configurations, for the shape signatures, are all using the nearest-neighbor decision rule. For *AREA* and *RADSD* feature spaces, in most of the cases, classifiers based on the average decision rule perform better. It is also evident from the summary tables that, for LANGG68 and ETH80 datasets, classifier configurations with *SSNRA* feature space outperform configurations with other feature spaces. While, for the COIL-100 and ALOI-1000 datasets, *SSH* feature space leads to the best performing base classifiers. In general, some of the best classification results are obtained with the *SSNRA* and *SSH* feature spaces and *NN* decision rule.

In the case of different shape signatures, with respect to different measures of similarity, *MPS* performed clearly above $P_\triangle$ and *ES* when the *AVG* and *NC* decision rules were used. Referring to Table C.1, in all the 24 experimental scenarios with these configurations, *MPS*

Table 7.4: Average performance of the AREA and RADSD feature spaces, similarity measures and decision rules using one-step learning.

	Average acc.(%)			
	LANGG68	COIL-100	ETH80	ALOI-1000
Similarity measure				
ES/MPS	23.9	7.4	35.5	10.8
$P_\triangle$	25.6	8.6	37.5	6.4
Feature space				
AREA	29.9	9.8	40.5	15.4
RADSD	19.7	6.2	32.5	1.8
Decision rule				
AVG	27.2	9.3	40.1	7.8
NC	24.8	8.3	36.1	8.5
NN	22.3	6.3	33.3	9.4

led to the highest classification accuracy. In the case of classifier configurations using *NN* rule, accuracy values are very similar in most of the cases (multi-resolution measures, *MPS* and $P_\triangle$ preform slightly better than *ES*). This suggests that some learning and classification strategies can suffer more from the 'matching by minimization' anomaly (described in Section 4.2.2) than others. Results from Table 7.3 reaffirm these conclusions and show that, over all datasets, *MPS* similarity measure, an original contribution of this thesis, outperforms both *ES* and $P_\triangle$. Also note, $P_\triangle$ performed the poorest in terms of average accuracy for all datasets. We believe that the poor performance of $P_\triangle$ is caused, to a large extent, by the 'matching by minimization' anomaly.

Any configuration involving uni-dimensional feature spaces (see Table 7.4), in general, performed poorly. Most notably, and in contrast to the results obtained with the shape-signatures, in almost all the cases, the configurations with $P_\triangle$ performed better than *ES/MPS*, and classifiers using *AVG* rule performed better than with *NN* rule. Having mentioned this, the overall poor performance of *AREA* and *RADSD* feature spaces does not allow us to make concrete conclusions with regards to the measures of similarity or the classification rules.

7.5 Results obtained with incremental learning

For classifiers trained using incremental learning, the different base classifier configurations and respective results are listed in Table C.2 and the summarized results are shown in Tables 7.5 and 7.6. The summary tables show the average accuracies and the average number of instances per category (stored during incremental training), for different shape signatures,

Table 7.5: Average performance of the different shape signatures, similarity measures and decision rules using incremental learning.

	LANGG68		COIL-100		ETH80		ALOI-1000	
	Avg. acc.	#inst/cat	Avg. acc.	#inst/cat	Avg. acc.	#inst/cat	Avg. acc.	#inst/cat
Similarity measure								
ES	60.3	45.7	50.8	58.4	54.3	50.1	42.5	65.6
MPS	**63.1**	**43.1**	**54.4**	**55.0**	**57.6**	**47.0**	**44.8**	**63.5**
P_Δ	58.6	46.9	48.4	60.4	53.7	50.1	39.1	68.9
Feature space								
SLH	60.5	45.4	50.3	57.4	54.0	49.5	36.6	69.8
SSH	56.6	50.2	**55.9**	**55.1**	55.5	48.8	**47.5**	**61.9**
SSNRA	**70.5**	**34.1**	53.9	55.3	**62.6**	**42.8**	46.6	62.5
SSNRSD	55.0	51.5	44.7	63.9	48.7	55.2	37.9	69.8
Decision rule								
AVG	52.1	51.9	30.7	71.9	50.2	52.9	20.7	81.3
NC	61.6	45.5	56.2	56.2	54.9	49.7	47.5	64.0
NN	**68.3**	**38.4**	**66.7**	**45.7**	**60.6**	**44.6**	**58.3**	**52.7**

similarity measures and decision rules.

The result analysis, carried out in the previous section, for the results obtained on classifier configurations trained using one-step learning, is equally valid for the configurations trained using incremental learning. Top classifier configurations, for different shape signatures, are obtained with *NN* rule. Configurations with *MPS* measure, in general, performed better than those with P_Δ and *ES*. The best classification results are obtained with *SSNRA* and *SSH* features spaces with *NN* decision rule.

The analysis derived from the average accuracies is also validated by the second evaluation metric. The least number of training instances were stored/required per category, for the best performing similarity measures, feature spaces and decision rules. That is, the best performing configurations, in terms of accuracy values, were also the best in memory usage. In both learning scenarios, one-step and incremental, the *NC* decision rule, which is a combination of *NN* and *AVG* rules, led to accuracy values lying between those obtained with *AVG* and *NN* rules.

When comparing Tables 7.3 and 7.5, some key differences stand out. It is clearly visible, for example, that the performance of classifiers using *NN* and *NC* decision rules deteriorates when training is incremental (this deterioration is greater for *NN* than *NC*). This result is not surprising, since, on average, for all *NN-INC* (*NN* classifiers that were trained incrementally) classifier configurations, less than 50% of the instances are used for making the classification decisions. Given that classification decisions are made with less information, the reduction in performance with respect to *NN-OS* is expected.

Table 7.6: Average performance of the AREA and RADSD feature spaces, similarity measures and decision rules using incremental learning.

	LANGG68		COIL-100		ETH80		ALOI-1000	
	Avg. acc.	#inst/cat	Avg. acc.	#inst/cat	Avg. acc.	#inst/cat	Avg. acc.	#inst/cat
Similarity measure								
ES/MPS	22.1	78.1	7.0	93.2	33.8	66.7	10.5	91.8
P_Δ	24.0	76.4	8.0	92.1	34.7	65.2	6.2	94.9
Feature space								
AREA	27.6	72.8	9.1	91.0	37.7	62.4	15.0	88.0
RADSD	18.4	81.9	5.9	94.2	30.8	69.4	1.8	98.7
Decision rule								
AVG	25.8	75.2	8.8	91.6	36.7	63.6	7.7	94.0
NC	22.4	77.4	7.8	92.5	34.0	66.2	8.6	92.7
NN	20.9	79.2	5.9	93.8	32.0	67.9	8.9	93.4

More surprising is the improved performance of *AVG-INC* with respect to *AVG-OS* classifier configurations. Comparing Tables C.1 and C.2, for the configurations with *AVG* rule, this improvement is also visible in almost all the cases and across all the datasets. Although, for LANGG68, COIL-100 and ALOI-1000 datasets, these improvements are relatively marginal, for the ETH80 dataset, the improvement is significant in all the cases. This marked improvement is especially reflected in the accuracy values for the configurations involving *SSNRA* feature space with the *AVG* decision rule.

There is a clear positive impact of incremental training on the classification performance for *AVG* classifiers. The reason for this improvement lies with the training procedure where only the "relevant" instances are stored to describe categories. Since the classification decision is arrived at by combining information from all the instances of a category, presence of misleading (diverging far from the average) instances will lead to poorer performance of *AVG* based classifiers (as noticed with *AVG-OS* classifiers). *NN* based classifiers, on the other hand, do not suffer from this drawback, because the classification decision using the *NN* rule is a local decision. Therefore, for *NN* based classifiers, presence of more instances leads to improved performance (as noticed with *NN-OS* classifiers).

7.6 Implications for the multi-classifier approach

As is evident from the results reported in the previous two sub-sections, for the shape signatures, irrespective of the decision rule, classifiers using the *MPS* measure consistently perform better than the ones based on *ES* or P_Δ. On the other hand, for the single dimensional feature spaces, the same is true for the measure based on Pyramid match, P_Δ. Therefore, for the learning approach based on multiple classifiers and meta-learning (MCML), the following base classifiers are used:

- 12 classifiers based on multi-dimensional feature spaces, where each classifier configuration consists of the feature space (*SLH/SSH/SSNRA/SSNRSD*), *MPS* similarity measure and a decision rule *NN/AVG/NC*;

- 6 classifiers based on single-dimensional feature spaces, where each classifier configuration consists of the feature space (*AREA/RADSD*), P_Δ similarity measure and a decision rule *NN/AVG/NC*;

In total, the learning architecture supported by MCML consists of 18 base classifiers, 4 classifiers based on Dempster-Shafer combinations (*DS2TOP, DS3TOP, DS4TOP* and *DS5TOP*) and 3 classifiers based on majority-voting (*MAJ3TOP, MAJ5TOP* and *MAJORITY-ALL*).

7.7 Evaluation of the word similarity measures

7.7.1 Isolated words dataset

To perform an independent evaluation of the two word similarity measures, *Edit distance* and *Phoneme-MFCC* measures presented in Section 6.2, a dataset of isolated spoken words was created. This dataset consists of names of 13 object category: *cup*, *star*, *jeep*, *scissor*, *car*, *horse*, *fork*, *stapler*, *knife*, *train*, *bike*, *boy*, *screwdriver*.

This dataset was collected with the help of 8 volunteers (5 males and 3 females), none of whom had English as their first language. Age of the participants ranged between 28 and 33 years. A small software program was developed which flashed the name of the word to be spoken and the participants were given a 2 seconds time-frame to speak each word. In total, 5 utterances per word per person were recorded. This led to a collection of 520 ($5 \times 13 \times 8$) isolated spoken words. Simultaneously, a text file was generated which contained the name (in text) and the location of each of the stored audio files.

7.7.2 Evaluation methodology and results

10-fold cross-validation was used to evaluate the performances of the similarity measures in a nearest-neighbor classification scenario. The dataset was evenly divided into 10 subsets of equal size, such that each subset contained equal number of instances of each of the 13 words. For each validation step, one of the subsets was taken out as the test/validation set and the word categories were represented by all the remaining instances. In each validation step, each instance in the test set was classified according to the nearest-neighbor rule.

For the classifier based on the *Phoneme-MFCC measure*, an average classification accuracy of 74.81% (±6.81) was achieved. This similarity measure was compared with *edit distance*

(Levenshtein distance) based metric, normally used in string matching algorithms (Navarro, 2001). Using this metric, the distance between two strings is computed as the minimum number of edit operations (insertion, deletion and substitution actions) required to transform one string into another. To use this metric, only the phoneme sequence is required to represent a word, discarding the MFC set. Using 10-fold cross-validation, the average classification accuracy of 80.96% (±6.81) was achieved.

These results indicate that, for comparing word representations, finding the cheapest path, *Edit distance*, may give a better performance than following the path centered around the diagonal elements *Phoneme-MFCC similarity.* Although the difference in performances is not huge, considering that Edit distance did not use any extra information (e.g. the MFC set), it can be a cheaper and likely a better alternative to our method.

7.8 Summary

This chapter focused on the classical evaluation of instance-based (Section 5.1) and cluster-based (Section 5.2) classifiers. Each classifier is a combination of a specific shape signature, a similarity measure and a classification rule. This led to a total of 48 classifiers to be evaluated.

Two learning approaches were explored: one-step and incremental. The evaluation of both approaches was carried out using 10-fold cross-validation. Moreover, the evaluation was carried out on four different object-images datasets.

From the three similarity measures used, classifier configurations using *Manhattan-Pyramid Similarity*, proposed in this thesis, consistently led to better performance. With respect to the shape signatures, configurations involving *Shape Slices Histogram* and *Shape Slices Normalized Radii Averages* led to better performance. From the decision rules explored, the *nearest neighbor* rule consistently performed better than the *nearest cluster* rule, which in turn performed better than the *average* classification rule.

Finally, on an isolated words dataset, two word similarity measures (Phoneme-MFCC measure and Edit distance) were compared in a nearest-neighbor classification scenario. The evaluation was carried out using 10-fold cross-validation. The reported results indicate that Edit distance resulted in better classification, using lesser information than the Phoneme-MFCC metric.

8 Open-ended evaluation

In the course of this thesis, several category learning and recognition approaches were developed to support the acquisition of vocabulary for object naming, with emphasis on online, incremental and open-ended learning. To evaluate such approaches, it is essential that the evaluation strategy gives explicit consideration to assessing the impact of the presentation of new categories, on the learning performance evolution.

However, for most of the word/category learning research, the set of categories is usually predefined (although, for exceptions, see our previous work (Seabra Lopes and Chauhan, 2007) and recent works of (Kirstein and Wersing, 2011; Kirstein et al., 2012) on lifelong learning). This is especially visible in the field of computer vision (Andreopoulos and Tsotsos, 2013; Everingham et al., 2010; Grauman and Darrell, 2007; Le et al., 2012; Uray et al., 2009). Even after the availability of large and challenging multi-class datasets, such as ALOI-1000 (Geusebroek et al., 2005), ImageNet (Deng et al., 2009), LabelMe (Russell et al., 2008) etc.), the standard direction is to develop algorithms that are designed for batch learning, whereas, online, incremental and open-ended learning is largely neglected. Given that the number of categories is usually predefined, the evaluation procedure follows the standard train and test approach.

In the field of language grounding, the emphasis is directly on the open-ended nature of word/category acquisition. However, due to the lack of suitable approaches to evaluate open-ended learning algorithms, this field of research has borrowed evaluation methods designed for standard multi-class evaluation (similar to the ones used in computer vision) (Chella et al., 2009; Kirstein and Wersing, 2011; Kirstein et al., 2012; Lovett et al., 2007; Roy and Pentland, 2002; Skočaj et al., 2007; Steels, 2003; Yu, 2005). Therefore, for evaluation purposes, in this case also, the target set of words is normally predefined. The evaluation methodology usually involves extracting certain measures, such as accuracy (Chella et al., 2009), semantic accuracy (Roy and Pentland, 2002), classification success (Steels and Kaplan, 2002), word-meaning grounding accuracy and object classification accuracy (Yu, 2005). Some authors plot this type of measures versus training time. As the set of words/categories is predefined, the plots usually show a gradual increase of these measures and a convergence to a 'final' value that the authors consider acceptable.

Having the set of words/categories fixed is contrary to the open-ended nature of the word learning domain. Standard methods for evaluating multi-class learning approaches cannot account for these aspects of open-ended learning. In this thesis work, a new generic *Teaching protocol* was designed to evaluate category learning and classification approaches that are open-ended (Section 8.1.2). Instantiations of the learning architectures described in Chapter 3 are then evaluated using this protocol (see Section 8.4.1). These evaluations are conducted using the datasets described in the previous chapter (see Section 7.2).

8.1 Teaching protocols for experimental evaluation

A well-defined protocol can facilitate the comparison of different approaches as well as the assessment of future improvements. With this in mind, and taking into account the evolution, recovery and breakpoint features (outlined in Section 3.1), a generic *Teaching protocol* (previously proposed in (Seabra Lopes and Chauhan, 2007)) was designed to evaluate open-ended category learning and classification approaches. In this section, after presenting the original protocol, we will discuss its limitations and propose an improved version.

8.1.1 Original teaching protocol

The original teaching protocol (Seabra Lopes and Chauhan, 2007), outlined in Algorithm 5, was designed to be applicable for any open-ended category learning domain. It is an elaborate and exhaustive evaluation procedure, where, for every new category introduced to the agent, the average protocol accuracy of the system is calculated by performing classification with all known categories.

To that end, the instructor repeatedly shows instances of the known categories, checks the agent's predictions and provides corrections when necessary. New (previously unseen) instances of the known categories are introduced to the agent, for classification, in the same sequence in which the categories were introduced. A subsequence of question/correction iterations in which instances of all the categories are shown to the agent is referred here as a "run". That is, given that n categories have already been introduced to the agent, a single run consists of n question/correction iterations. In each question/correction iteration, a new

Algorithm 5 Original teaching protocol (Seabra Lopes and Chauhan, 2007)

```
parameter:
    $A_{min}$        ▷ Accuracy threshold, by default set to 0.667
Introduce $Category_1$;
$n \leftarrow 1$
repeat
    $n \leftarrow n + 1$                                        ▷ Ready for the next category
    Introduce $Category_n$;
    $k \leftarrow 0$
    $c \leftarrow 1$
    repeat
        Test and correct the learning system by presenting a
            previously unseen instance of $Category_c$
        $c \leftarrow c + 1$
        if $c > n$ then $c \leftarrow 1$
        if $k \leq 3n$ then $k \leftarrow k + 1$
        $A \leftarrow$ average accuracy in last $k$ question/correction iterations  ▷ Protocol accuracy

    until (($A > A_{min}$ and $k \geq n$) or                 ▷ Accuracy threshold crossed
          (user sees no improvement in protocol accuracy))   ▷ or Breakpoint reached

until (user sees no improvement in protocol accuracy)        ▷ Breakpoint reached
```

instance for one of the known categories is shown to the agent for classification (and correction is given if the agent's classification is incorrect).

The primary metric controling the flow of application of the protocol is *protocol accuracy*, which is an accuracy value calculated over the latest classification results. When the number of classification results since the last time a new category was introduced, k, is greater than or equal to n but less than $3 \times n$, all results are used to compute the protocol accuracy. The criterion that indicates that the system is ready to accept a new category is based on the accuracy threshold, A_{min}. In case the protocol accuracy fails to reach the accuracy threshold after a certain number of iterations, the user can infer that the agent is no longer able to learn more categories and the *breakpoint* has been reached. Since the protocol was designed to be followed by a human user (non-automated, manual interaction), the recognition of the breakpoint is subjective. When the user decides that the agent's learning capabilities have reached the breakpoint, the protocol is stopped.

It should be noted that the protocol accuracy, as described above, is an external measure that controls the application of the teaching protocol for experimental evaluation. It should not be confused with internal measures such as the *classifier success*, that is used by the multi-classifier approach (Section 5.4). The classifier success measure is an internal metric computed by the agent and mainly used to drive classifier combinations. The main difference between both is that, whereas classifier success takes into account all classifier results since the agent was running (giving more weight to more recent results), the protocol accuracy measure ignores any results given by the agent before the last time a new category was introduced. This is because the protocol accuracy is used to analyze the impact of the introduction of a new category, on agent's learning performance, from initial instability to recovery.

8.1.2 A generalized protocol for automated experiments

The original teaching protocol is sufficient to evaluate open-ended learning approaches, however it is extremely exhaustive. In particular, the protocol requires that the user, in each run, shows instances from all the known categories. Testing each category, in each run, is perhaps unnecessary and leads to a greater number of question/correction iterations than needed. A smaller number of iterations - testing a smaller, randomly chosen subset of the known categories - could suffice to reach the same conclusions. This will lead to less time spent in evaluation. With this in mind (and a few other factors, as discussed below), a new teaching protocol is proposed which is a significantly improved version of the original protocol.

The primary difference between the old and the new protocols is the number of instances shown in a single run. In the original protocol, each of the n known categories is tested once per run. In the new protocol, it is enough to test a subset of all known categories in each run. The number of categories to test in a run, and thus the 'number of iterations per run', R, is given by the following formula:

$$R = round\left(\frac{n}{1 + f \cdot n}\right) \tag{8.1}$$

where, n is the number of categories known to the agent, and $f \in [0, 1]$ is the run reduction factor. The denominator on the right-hand side of the equation is a linear function of n, with the y-intercept at 1 and the slope f. f controls the speed at which the denominator grows as a function of the n. A larger value of f leads to smaller numbers of iterations in each run. At the higher extreme, if $f = 1$, a single category will be tested in a run (that is, one iteration per run). At the other extreme, when $f = 0$, instances from all the known categories will be shown to the agent in each run. Thus, in this respect the new protocol generalizes the original teaching protocol, that is, the original protocol is a special case of the new one.

Figure 8.1 shows the progression of R, for different values of f, with respect to the number of categories. It can be observed from this figure, for $f \geq 0.1$, R converges to values below 10. That is, as the number of categories grows, the number of tests in a single run will never reach beyond 10. Bigger values of f will lead to even fewer number of categories being tested in each run. For $f = 0.01$, the number of question/correction iterations per run continues to grow steadily. At the introduction of the 1000-th category, 91 categories will be tested in each run. For the values of f between 0.05 to 0.15, R converges towards 20 and 7 iterations per run, respectively. For the experiments reported in this chapter, by default, f was set to

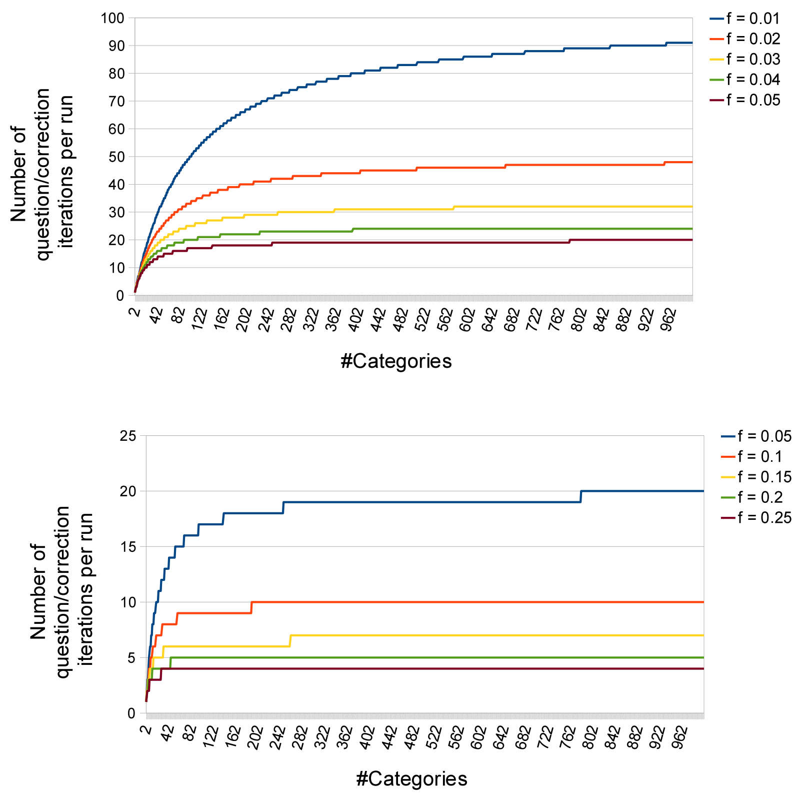


Figure 8.1: Evaluation of the 'number of iterations per run', R, as the number of categories increases from 1 to 1000, for different values of 'run reduction factor' f (in the range [0.1, 0.25]). To clearly visualize the change in R, for different values of f, the plot is divided in two.

0.05.

The second difference, with respect to the original protocol is concerned with the sequence in which categories are tested in each run. In the original protocol, categories are tested in the sequence in which they were introduced to the agent. This eventually leads to more instances being stored for the categories introduced earlier. In the improved protocol, categories are tested in a random sequence.

As in the original protocol, the protocol accuracy is calculated over the latest runs of question correction iterations. When the number of classification results since the last time a new category was introduced, k, is less than $3 \times R$, the average of all results is used. The criterion that controls whether a new category is to be introduced to the learning agent, as in the original protocol, is based on the accuracy threshold. More specifically, at least one run must be complete ($k \geq R$) and the protocol accuracy must be equal or higher than the threshold ($A > A_{min}$).

The breakpoint is reached when the agent is no longer able to reach the accuracy threshold, after a new category is introduced to the agent. Declaring the breakpoint, therefore, is based on testing the learning performance of the agent for a "sufficient" number of question/correction iterations, without reaching the accuracy threshold. In the original protocol, the breakpoint criterion was a subjective decision of the human user. Subjectivity implies that the decision of reaching the breakpoint can potentially vary from one experiment to another and from one user to another. A solution, implemented in some of the previous works (e.g. Chauhan and Seabra Lopes, 2010a; Seabra Lopes and Chauhan, 2007, 2008), for semi-automated experiments, was to put a hard limit of 50 runs, *i.e.* $50 \times n$ question/correction iterations, where n is the current number of known categories, to recognize the breakpoint. Using this criterion, as the agent begins to learn a larger number of categories, the number of iterations required before declaring breakpoint becomes extremely inflated[1].

For the new protocol, a breakpoint criterion was designed which is a function of the number of categories learned by the agent. Here, given that n categories have been previously introduced to the agent, the number of question/correction iterations with protocol accuracy below the accuracy threshold, after which breakpoint is declared, is computed as:

$$B = n \cdot (1 + P) \tag{8.2}$$

where

$$P = p \cdot exp^{-n \cdot d} \tag{8.3}$$

describes the "teacher's patience" as a function of n, the number of learned categories. This patience value is a decay funtion used to control B. Here, the parameter p is the *initial patience* value and the *patience decay factor*, d, controls the rate at which P decays with respect to n. Figure 8.2 plots B for different values of d, where $p = 20$. The behavior of B can be dissected into three phases:

– First phase starts at the beginning of the application of the protocol, when only few categories have been introduced to the agent, where B increases towards a peak value. The idea is that, for small number of categories, the number of evaluations (*i.e* the "sufficient" number of question/correction iterations before the breakpoint is declared) should be much greater than the number of known categories.

– The second phase is identified by the period where B decays with respect to n. The number of question/correction iterations should not continuously increase with the number of categories (this was one of the main problems with the old protocol). The strong increase in B in the first phase gets attenuated in the second partition. This segment

[1]As an example, consider that after introducing the 300th category, the learner is no longer able to reach the accuracy threshold. Using the old protocol, the number of iterations required before the breakpoint is declared will be 15,000.

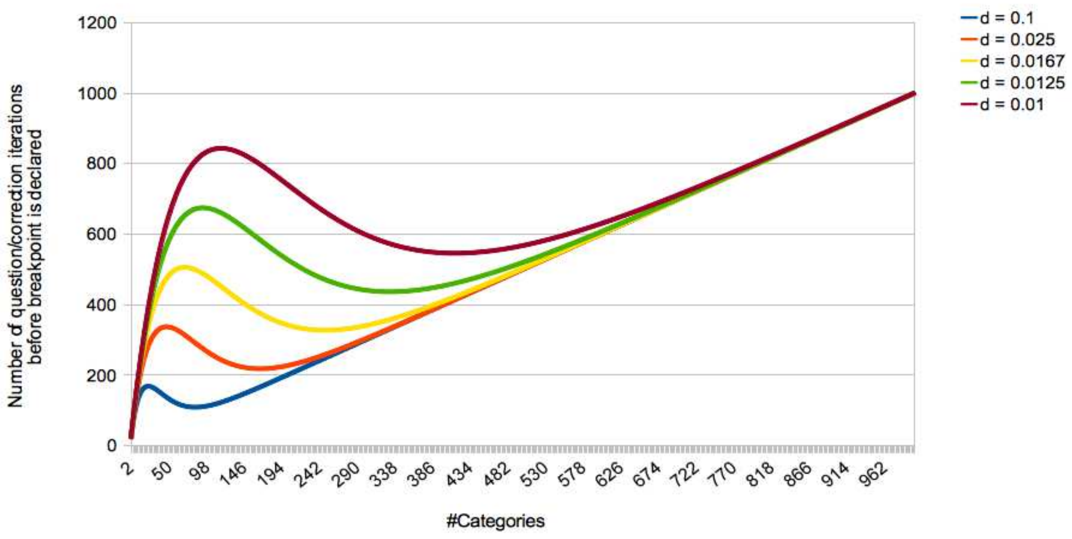


Figure 8.2: Evolution of the 'number of iterations before breakpoint is declared', B, with "initial patience" $p = 20$, and for different values of the patience decay factor, d, as the number of categories increases from 1 to 1000.

continues until P begins to approach 0.

– Third phase begins when P approaches 0. Here onwards, B approximates to n.

In the performed experiments, $p = 20$ and $d = 0.01$ were used. With these parameters, the result is that after introducing the 2^{nd} category, *i.e.* $n = 2$, 20 iterations are allowed before breakpoint is declared. Similarly, following the mentioned formula, when $n = 10, 500$ or 1000 will result in $B = 190, 567, 1000$, respectively. The accuracy threshold was set to 0.667 in the experiments reported. This threshold ensures that, in a stable situation, there will be at least twice as many correct answers as incorrect answers, which intuitively appears to be a suitable baseline for acceptable performance.

8.2 Simulated teacher

Having a human teacher to follow the teaching protocol (either old or new) is highly time consuming and exhausting. From the teacher's perspective, following the protocol entails showing objects to the agent, teaching category names, requesting classification and providing corrective feedback. Each of these tasks consumes some time and, during a protocol-directed experiment this time accumulates and a single experiment can take days or weeks to finish. One such experiment, reported in our previous work (Seabra Lopes and Chauhan, 2008), took more than a week to complete. Here, the agent was able to learn 68 categories in 3767 question/correction iterations (which approximates to 540 iterations per day or, on an

Algorithm 6 New teaching protocol

```
parameters:
    $f$              ▷ Controls the number of question/correction iterations per run.
                       By default, $f = 0.05$
    $A_{min}$        ▷ Accuracy threshold, by default set to 0.667
    $p$              ▷ Initial patience factor. By default, $p = 20$
    $d$              ▷ Patience decay factor. By default, $d = 0.01$
Introduce $Category_1$;
$n \leftarrow 1$
repeat
    $n \leftarrow n + 1$                                    ▷ Ready for the next category
    Introduce $Category_n$;
    $R \leftarrow \text{round}(n/(1 + f \cdot n))$
    $B = n \cdot (1 + p \cdot exp^{-n \cdot d})$            ▷ For Breakpoint criterion
    $I \leftarrow$ array containing all category indices $(1, ..., n)$, so far
    $k \leftarrow 0$
    $c \leftarrow 1$
    repeat
        if $c = 1$ then Randomly rearrange $I$
        Test and correct the learning system by presenting a
            previously unseen instance of $Category_{I[c]}$
        $c \leftarrow c + 1$
        if $c > R$ then $c \leftarrow 1$
        if $k \leq 3R$ then $k \leftarrow k + 1$
        $A \leftarrow$ average accuracy in last $k$ question/correction iterations  ▷ Protocol accuracy
    until (($A > A_{min}$ and $k \geq R$) or                 ▷ Accuracy threshold crossed
          ($k \geq B$))                                      ▷ or Breakpoint reached

until ($k \geq B$)                                           ▷ Breakpoint reached
```

average, 10 categories taught per day). Examples of similar experiments, where a human teacher follows the protocol, have been reported in our other works as well (see e.g. Chauhan and Seabra Lopes, 2010a,b; Chauhan et al., 2009; Seabra Lopes and Chauhan, 2007; Seabra Lopes et al., 2007).

Therefore, a simulated "teaching agent" (or simulated teacher) was developed which takes over the task of the human teacher as the agent's instructor. The simulated teacher was designed to follow the teaching protocol and is capable of interacting with the learning agent using the *teach, ask* and *correct* actions provided by the human-robot interaction interface (as described in Section 3.2).

Given an object-images dataset, the simulated teacher picks images randomly for interaction with the learning system and uses each stored image at most once. Following the new teaching protocol, when the agent is ready to learn a new category, the next category is selected randomly. When the simulated teacher runs out of images for a particular category, the human teacher can be called to interact with the agent, so that the experiment can continue with its normal course. Each time the simulated teacher calls the human teacher for interaction with the agent, additional images are collected and stored, which will be used in later experiments. The functionality to allow the teacher to add new images is part of the normal functionality of the teaching agent and has been used in previous works (e.g. Chauhan and Seabra Lopes, 2010a; Seabra Lopes and Chauhan, 2008). However, this functionality cannot be employed when working with publicly available datasets and therefore not utilized for the experiments reported in the thesis.

Replacing the human teacher with a simulated one makes it possible for us to conduct systematic experiments that would, otherwise, take many weeks to complete. More importantly, having a simulated teacher allows the possibility to perform multiple experiments and explore different experimental conditions in a fraction of time a human would take to carry out the same task.

8.3 Evaluation measures

As mentioned earlier, the protocol accuracy measure, in combination with the accuracy threshold, controls the application of the teaching protocol. *Evolution, recovery* and *breakpoint* features, presented in Section 3.1, are reflected by this measure over successive question/correction iterations. The change in the values of this measure, from the iteration where a new category is introduced till the iteration where the accuracy threshold is reached, shows the evolution of the agent's learning performance as it adjusts the category descriptions to accommodate the new category. Additionally, the total number of iterations required to reach the accuracy threshold, in this context, shows the period of recovery. Breakpoint is reached when, after a sufficient number of question/correction iterations, the learning agent is no longer able to accommodate the new categories.

Once an experiment concludes, additional criteria and measures are used to evaluate the overall learning performance of the agent during an experiment. The following measures characterize the quality and coverage of the learned knowledge after an experiment has finished:

– **Global accuracy**: This is given as the percentage of correct predictions made during a complete experiment. This measure gives the overall performance of the agent during an experiment;

– **Average protocol accuracy**: The average of all protocol accuracy values, obtained during the application of the protocol, computed over all the question/correction iterations; and

– **Number of learned categories** during the experiment.

For characterizing the learning process in terms of space/time resources, the following measures are used:

- **Number of question/correction iterations** during the experiment;
- **Average number of instances stored per category** throughout the experiment.

Each of the category learning architectures, described in Section 3.3, is evaluated at the task of open-ended word/category learning using the new teaching protocol and the evaluation measures presented here.

8.4 Grounding textual words

The learning approaches implemented on the three architectures that ground textual vocabulary are compared in this section. The learning behavior of the architecture supporting multiple classifiers and meta-learning is studied across a long-term experiment in sub-section 8.4.2. To evaluate the contribution and impact of meta-learning, different meta-learning configurations are compared in the sub-section 8.4.3.

8.4.1 Evaluation and comparison of different learning approaches

Three category learning architectures were designed for grounding textual vocabulary, and each of these architectures differs in the learning and classification approach used. Three instantiations of these architectures are evaluated in this section. Each architecture and the corresponding learning approach is listed here:

- Architecture based on instance-based learning with a single classifier:

  [**COMP**] - Learning approach is instance-based where categories are represented by the graph of components of individual instances (described in Section 4.1.2) and classification is achieved using a classifier based on the nearest-neighbor rule (see Section 5.1.2). This classifier uses the graph of components similarity measure described in Section 4.2.4.
- Architecture based on one-class learning:

  [**SVDD**] - Learning and classification approach is based on genetic SVDD (described in Section 5.3.3). Here, the SSNRA feature space (described in Section 4.1.1) is used for representing objects.
- Architecture based on instance-based learning with multiple classifiers and meta-learning:

  [**MCML**] - Learning and classification is based on using multiple base-classifiers and classifier combinations and a meta-cognitive component. The meta-cognitive component maintains updated success statistics for all the classifiers, for dynamically reconfiguring the classifier combinations, and manages memory using a forgetting rule (described in Sections 5.4 and 5.5).

The objective of the experiments reported here is to evaluate the performance of these learning approaches with respect to open-ended category learning and vocabulary acquisition. Each of these approaches is evaluated using the new teaching protocol and the measures presented in the previous section.

Although our specific experimental scenario (explained in Section 3.5) allows a human user to show new objects to the agent in a relatively uncontrolled environment, in absence of comparison with other scenarios we run the danger of researcher bias (Onwuegbuzie and Leech, 2007). Therefore it becomes essential that the learning performance of the agent is evaluated on objects shown in conditions that are not controlled by us (*i.e.* the experimental conditions cannot be considered biased). This is achieved by performing experimental evaluation over all four object-images datasets described in Section 7.2.

The protocol is controlled by the simulated teacher, as discussed in Section 8.2. During the course of the experiment, the protocol accuracy measure is used to analyze the impact of the introduction of a new category on a learning system, from a possible initial instability to the final recovery. Breakpoint is reached when the learning system stops showing signs of evolution or recovery. After an experiment concludes, further evaluation is carried out to evaluate overall performance. Using images from a dataset implies that, when following the protocol, none of the images can be used more than once. In this scenario, it is possible that all the images from a specific category get exhausted before the experiment concludes. In such event, it is no longer possible to continue the protocol and the evaluation process is halted. In the reported results, this is shown by the stopping condition, "Lack of data". In this case, the results from the protocol are analyzed up to the iteration just before the latest category was introduced.

A single experiment involves the learning agent, supported by a specific learning architecture and the corresponding learning approach, and the simulated teacher. The teacher interacts with the learning agent by following the teaching protocol. For each experiment, one of the four datasets (detailed in Section 7.2) is used by the teacher as the source of new categories and object instances. An experiment is considered finished when either of the following conditions is met: *breakpoint* is reached; or all the images of a specific category have been used (*Lack of data*); or all the categories in the dataset have been learned (*All cats. learned*).

For each learning approach and dataset, five experiments were carried out[2]. The results from these experiments are summarized in Table 8.1.

Table 8.1: Summary of experiments for different open-ended learning architectures

Dataset	Learning approach	Stopping condition	# exps.	#cats learned	#iters	#inst/cat	Avg. protocol acc.(%)	Global acc.(%)
LANGG68	SVDD	Lack of data	5	25.0	1070.4	19.5	53.2	56.7
	COMP	Lack of data	2	14.5	761.5	26.1	50.4	52.2
		Breakpoint	3	8.3	140.3	8.22	52.5	60.4
	MCML	All cats. learned	5	68.0	855.6	3.8	77.0	78.1
COIL-100	SVDD	Lack of data	3	9.0	322.0	19.9	46.0	49.6
		Breakpoint	2	7.5	141.5	10.0	47.9	53.9
	COMP	Lack of data	4	17.5	530.0	13.3	54.7	58.6
		Breakpoint	1	10	158.0	7.5	48.5	57.6
	MCML	All cats. learned	1	100.0	2319.0	8.3	63.2	68.4
		Lack of data	4	64.3	1385.0	7.8	63.6	68.1
ETH80	SVDD	Breakpoint	5	3.6	35.0	5.4	49.9	59.0
	COMP	Breakpoint	5	4.2	66.4	8.1	46.1	50.4
	MCML	All cats. learned	5	8.0	141.6	8.3	53.7	59.9
ALOI-1000	SVDD	Breakpoint	5	3.4	54.2	8.5	41.3	47.3
	MCML	Lack of data	5	259.4	7789.0	10.6	64.4	68.0

In general, over all evaluation metrics, and across all the datasets, the MCML approach very clearly outperforms the other two approaches. For each dataset, in comparison to SVDD and COMP approaches, the MCML based approach learned the highest number of categories. In addition, with respect to the number of categories learned per dataset, MCML consistently stores the least number of instances (making this method the most memory efficient), and is the most efficient learner (for every dataset, the least number of question/correction iterations were required for the number of categories learned). For the longest experiment (on ALOI-1000 dataset), MCML was shown to learn 293 categories and the experiment was stopped when all the images of one of the categories had been used (indicating the potential for learning many more categories). The average protocol accuracy and the global accuracy of the MCML learner are also considerably higher, when compared with SVDD and COMP approaches. It should be noted that these results should be seen in the light of the number of categories learned. For example, in the experiments on ETH80 dataset, the SVDD and the MCML approaches seem to indicate similar global accuracy, however, SVDD reached the breakpoint after the introduction of the 6th category (*i.e.* 5 categories were learned), whereas

[2]The learning approach based on graph of components, COMP, was not evaluated on the ALOI-1000 dataset. The reason being, the method to extract components is designed for object images where the components to be found are inside the object boundary. If complete object boundary is not available, as is the case with a lot of [noisy] images in the ALOI-1000 datset, it is not possible to extract the components of that object

MCML learned all the 8 categories. Finally, for the LANGG68 and the ETH80 datasets, MCML allowed the agent to learn all the categories in these datasets, and for the COIL-100 and ALOI-1000 datasets, experiments were stopped when all the images of a certain category were exhausted (in one experiment on COIL-100, all the 100 categories were learned).

On the LANGG68 dataset, the approach based on SVDD performed better than the one based on COMP. In fact, for none of the experiments on this dataset the breakpoint was reached, suggesting the possibility that more categories could be learned. The main innovation in this approach is the use of genetic optimization of SVDD parameters (see Section 5.3.3). This optimization method was designed to improve over the standard (quadratic) optimization method that comes with the original implementation of SVDD (Tax, 2001). In our previous work (Seabra Lopes and Chauhan, 2007), an open-ended learning architecture, OCLL (briefly described in Section 3.3.2), was designed using the original SVDD implementation. Using the OCLL architecture, the agent was never able to learn more than 11 categories (breakpoint was reached) from the same images that are stored in the LANGG68 dataset. Whereas, using genetic SVDD, on the LANGG68 dataset, the agent was shown to learn an average of 25 categories (in one experiment, the agent learned 37 categories), and the breakpoint was not reached in any of the 5 experiments. For these experimental conditions, it can be concluded that the genetic optimization process does lead to a better incremental and multi-class learner when compared with SVDD using standard approach to optimization. Similar results have been reported previously on non-incremental one-class classification using SVDD where genetic optimization lead to better category descriptions when compared with the quadratic optimizers (Tavakkoli et al., 2007).

Referring back to the Table 8.1, it can be seen that the performance achieved by the SVDD approach on LANGG68 dataset does not repeat for any other dataset. In fact, for the other three datasets, the performance is clearly quite poor. The underlying reason for the drop in performance is the nature of these datasets. LANGG68 is the most homogeneous dataset where object images of the same category are taken in the same pose. In comparison, in other datasets, images belonging to a single category range from less homogeneous (COIL100), to noisy (ALOI-1000), to heterogeneous (ETH80). Within the scope of the results achieved here, the approach based on support vector data descriptions appears unable to cope with the incremental multi-class scenarios where categories are noisy and non-homogeneous.

COMP architecture, although better than SVDD in some cases, also leads to relatively poor overall performance, even in the case where categories consist of highly homogeneous object images (*i.e.* the LANGG68 dataset). One of the central aspects of this approach is the pre-processing step involving identification and merging of components. It would be expected that the final number of extracted components for the images belonging to the same category in the LANGG68 will be very similar. This is because the images from the same category were taken from the same viewpoint. However, the results obtained from the experiments

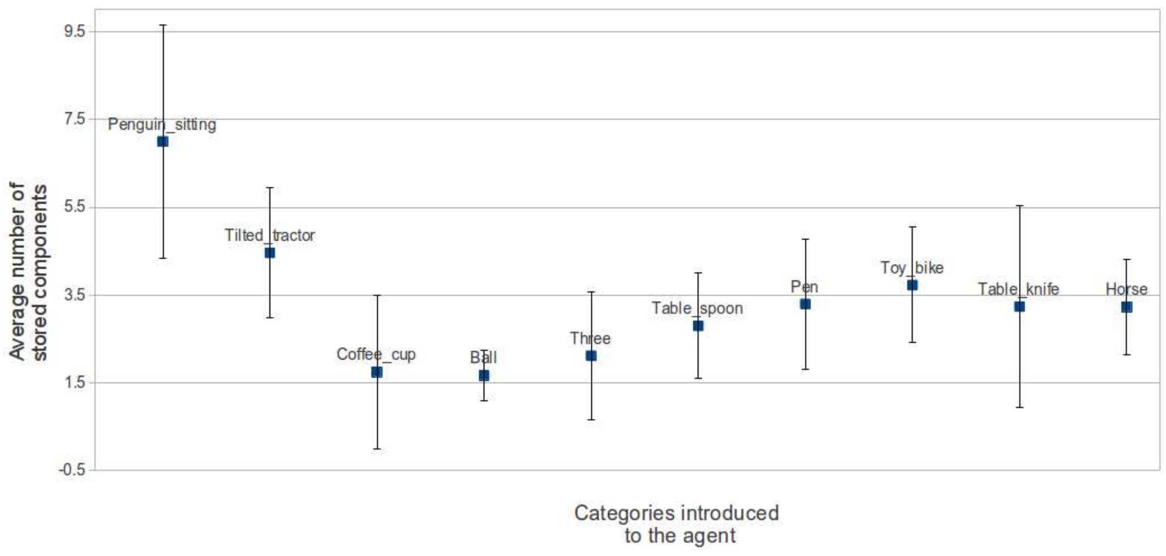


Figure 8.3: Average number of extracted components for the instances representing each category at the end of one experiment with COMP on LANGG68 dataset where the learner reached the breakpoint after introducing the 10th category. Error bars give the standard deviation computed over the number of components for the instances representing the categories.

show that the number of components that describe instances from the same category vary greatly. Figure 8.3 shows the average number of components and the corresponding standard deviations for the instances stored for each category at the end of one of the experiment on LANGG68 dataset. As an example consider the categories "Penguin_sitting", where, for the 28 stored instances, the number of extracted components range from 1 to 13, and "Table_knife", where, for 21 instances, components range from 1 to 11. The same is true for the other categories. Since component extraction is based on color information, the illumination changes have a significant impact on locating these components. The strategy to find components is not robust against illumination changes and this is the likely cause for the poor performance of COMP based approach.

Until now the discussion focused on evaluation of the learning approaches after an experiment has finished. This evaluation is computed over all the test iterations and hides the *evolution* and *recovery* process after the introduction of new categories. Further insight can be obtained by looking at the progress of protocol accuracy when applying the teaching protocol.

Figure 8.4 presents the evolution of protocol accuracy measure plotted against the test iterations for the experiment discussed in the previous paragraph. This experiment continued for 338 question/correction iterations and the agent was able to learn the 9 categories (breakpoint reached after the introduction of the 10th category). In general, protocol accuracy degrades after the introduction of each new category, then eventually recovers. The points where the values in the graph are either 100 or 0 are an indicator that after the introduction

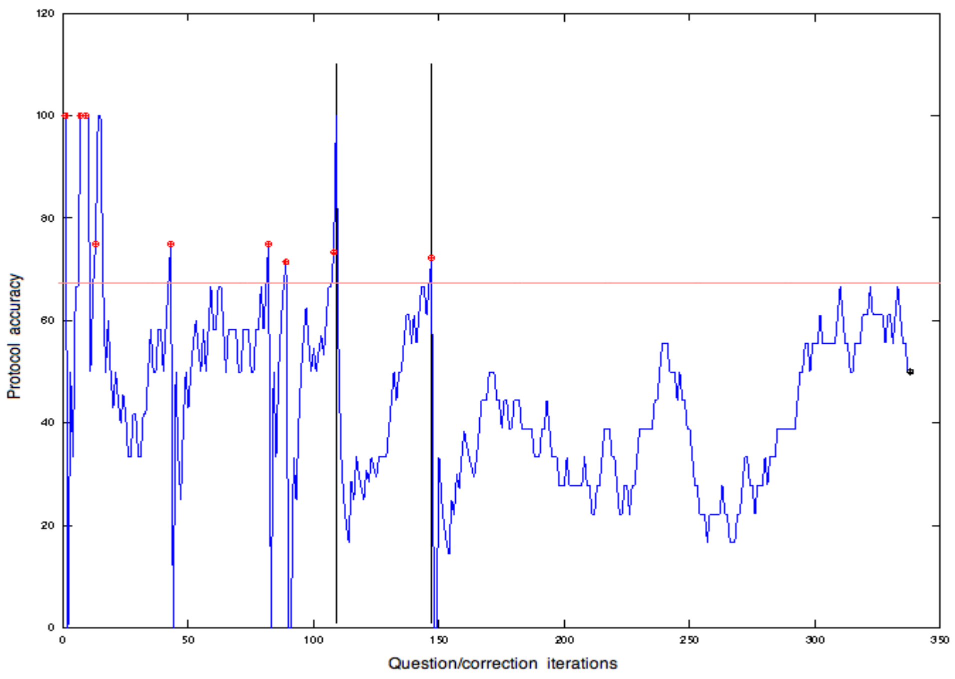


Figure 8.4: Evolution of protocol accuracy versus the number of question/correction iterations for a accuracy threshold of 66.67% (marked by the red horizontal line) for one of the experiments using COMP on the LANGG68 dataset. The circular markers highlight the iteration at which, after the introduction of a new category, the accuracy threshold was achieved. Breakpoint was reached after the introduction of the 10^{th} category.

---

of a new category, the first category prediction was either correct or incorrect. Towards the limit of the category discrimination abilities of the agent, learning starts to take longer. From Figure 8.4, we see that most categories (exactly 8 categories) were learned in the first 109 iterations (14 iterations per category), whereas learning the 9th category took ~40 iterations. The breakpoint is also clearly visible in this figure. After the introduction of the 10^{th} category, the protocol continued for 190 iterations, but the accuracy threshold could not be crossed. The breakpoint was declared (based on the criteria defined in eqn. 8.2) after iteration 338.

In all experiments, across all learning approaches, the protocol progresses in a similar fashion. The introduction of a new category normally leads to deterioration in classification accuracy, followed by a period of gradual recovery. In general, the introduction of a new category causes confusion in the prediction of existing categories, hence reducing the protocol accuracy of the system. Each incorrect prediction will lead the teacher to send a correction. This process is continued till the protocol accuracy reaches the accuracy threshold (unless it is already above the threshold). The key difference between different learning approaches is the amount of time spent in recovery before the threshold is reached, which is also an

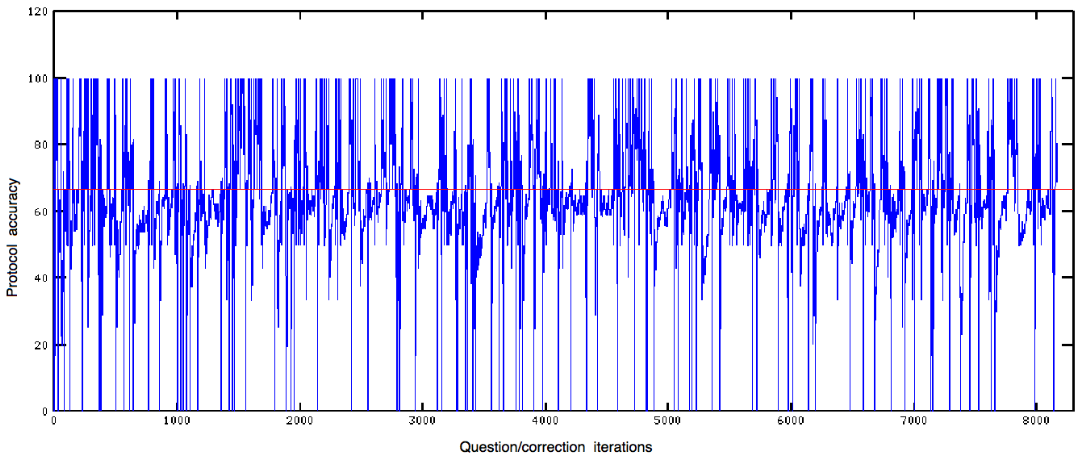


Figure 8.5: Evolution of protocol accuracy versus the number of question/correction iterations for a long-term experiment using MCML on the ALOI-1000 dataset. The agent successfully learned 293 categories and the experiment concluded prematurely due to the "Lack of data" criterion.

indicator of the learning performance. COMP and SVDD show a very similar pattern (as noticed in Figure 8.4) where, towards the end of the experiment, the recovery period grows substantially. On the other hand, for the MCML approach, in all experiments, the agent shows rapid recovery (see Figure 8.5 for one such example).

As mentioned earlier, of all the learning approaches, MCML led to the most successful category learning architecture. This approach comprises multiple base-classifiers and combinations derived over them. To further investigate the learning behavior of this approach, the next section is dedicated to a detailed analysis of different base classifiers over the longest experiment.

8.4.2 Detailed analysis of MCML over a long-term experiment

In the longest experiment with the multi-classifier approach, the agent successfully learned 293 categories. This experiment was conducted on the ALOI-1000 dataset and the teaching protocol halted because all the images from one of the categories had been used. In all, it took 8165 question/correction iterations for the agent to acquire these categories. Note that, using the new protocol, the minimum number of iterations required to learn this many categories is 4,774 (and the old protocol would have, at the least, taken 43,071 iterations). The results achieved at the conclusion of the experiment show the global accuracy of 68.7% and the average protocol accuracy of 65%. During the teaching/learning process, the agent stored a total of approximately 2860 training instances, leading to 9.8 instances per category.

Figure 8.5 plots the evolution of protocol accuracy against the number of question/ correction iterations. As observed for the experiment discussed in the previous sub-section, sections of the curve with more pronounced oscillations indicate the introduction of new categories (although it is difficult to visually notice the recovery curves for a long experiment). In general, protocol accuracy degrades after the introduction of each new category, then eventually recovers. It can be observed that as the number of categories grows, the period of recovery does not show a great increase. This indicates that the agent is still capable of learning more categories and its learning capacity is far from reaching the breakpoint.

A separate analysis of the classifiers included in MCML is important for assessing their individual contributions. Table 8.2 shows the averaged summary of the *successes* of all the classifiers during the complete experiment. As described in Section 5.4.1, this success measure is a classifier specific self-evaluation measure used in MCML. For each classifier, the success measure is an internal measure that takes into account its classification results since the agent was running, giving more weight to the more recent results.

To complement this summary, Figure 8.6 displays the learning curves for all classifiers. The classifiers based on the single-dimensional feature-spaces show the poorest performances. As the agent is shown more categories, there is a steep drop in classification success of the classifiers based on *AREA* and *RADSD* feature spaces. Amongst the base-classifiers, *SSH* and *SSNRA* based classifiers perform the best. The classifier configurations, *SSH-MPS-NN* and *SSNRA-MPS-NN*, are the top performers. Overall, with respect to the decision rules, *NN*-based classifiers outperform *AVG* and *NC* rules for the multi-dimensional feature spaces. For the single dimensional feature spaces, classifier configurations with the *AVG* rule perform better. These results relate very well to the separate evaluation of the base-classifiers in Section 7.3, where similar conclusions were reached.

The results from the dynamically reconfigured classifiers, which combine decisions from the base classifiers, show the best performance. Specifically, a classifier based on Dempster-Shafer approach that combines evidence from the top 5 base classifiers, *DS5TOP*, achieves an average success rate of approximately 66%. On average, *DS5TOP* is the best classifier throughout the experiment, closely followed by *DS4TOP* (65.7%) and *MAJORITY-ALL* (65.5%) which takes a majority vote over all the classifiers. Finally, the externally observable performance of the agent was 66.4%. It is also important to mention here that for the rest of the experiments, with the MCML approach, on all the datasets, very similar results were obtained. The conclusions drawn on the current experiment are directly applicable to the rest of the experiments. A summary of results over each dataset is provided in the 5^{th} row of the Table C.3 in Appendix C.

The significant improvement on learning performance of classifier combinations over base classifiers is clearly visible in the Table 8.2 as well as in the Figure 8.6. This is unsurprising and many works in literature have shown that classifiers that combine information from multiple weak classifiers, where the weak classifiers possibly use different feature-spaces, normally lead to much better performance (see e.g. Duin and Tax, 2000; Kittler et al., 1998).

As mentioned in Section 5.4.1, the meta-level component monitors classifier performance

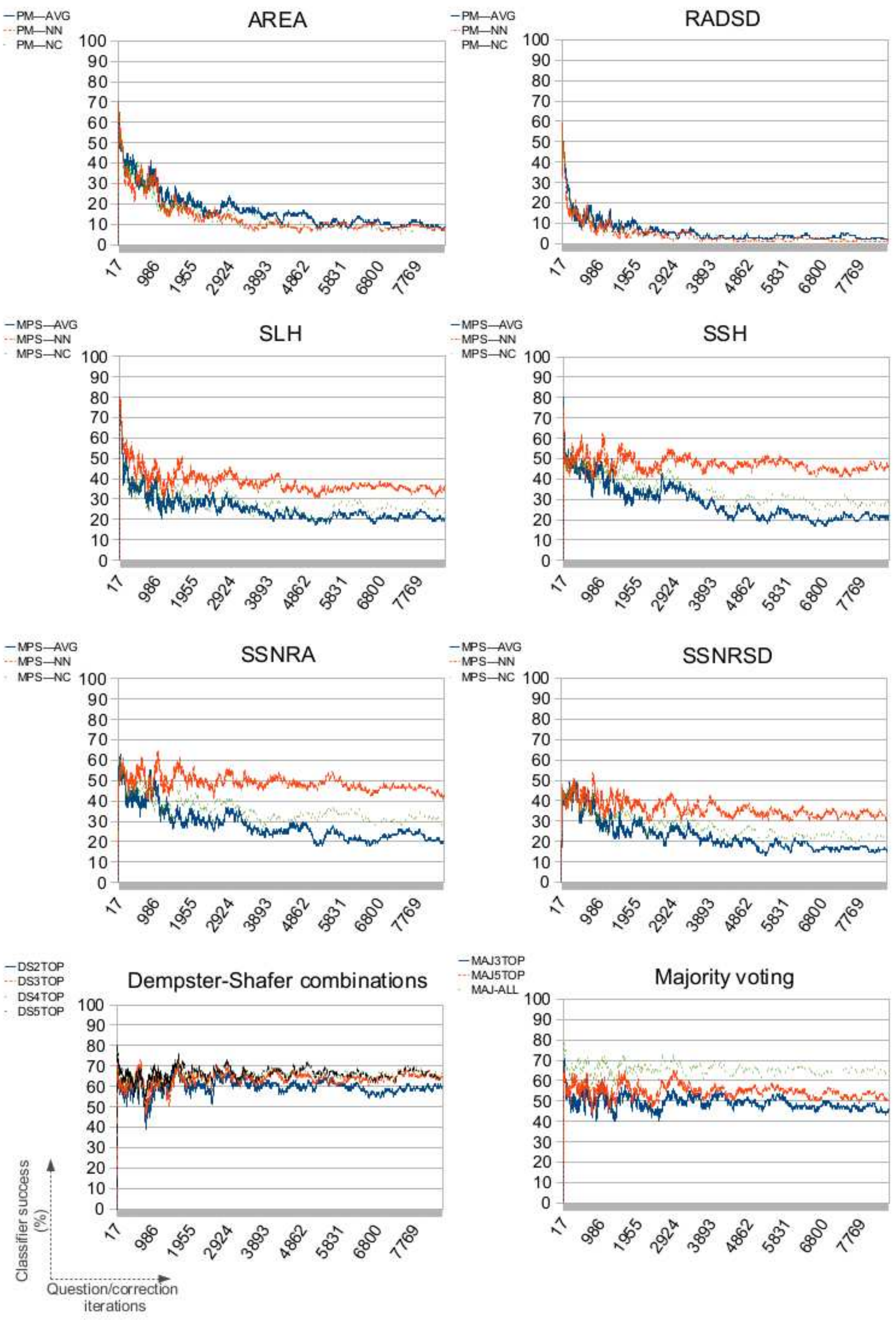


Figure 8.6: Evolution of success rate of each classifier with respect to the number of test iterations. In the top three rows, classifiers are organized by feature-space. The bottom row is organized by the type of classifier combination.

Table 8.2: Average classification success rates for all classifiers for the long-term experiment

Single dimension classifiers	
AREA-PM-AVG	16.8
AREA-PM-NN	13.6
AREA-PM-NC	13.7
RADSD-PM-AVG	6.1
RADSD-PM-NN	4.3
RADSD-PM-NC	5.1
Classifiers based on shape signatures	
SLH-MPS-AVG	25.4
SLH-MPS-NN	38.5
SLH-MPS-NC	27.7
SSH-MPS-AVG	28.7
SSH-MPS-NN	47.3
SSH-MPS-NC	34.1
SSNRA-MPS-AVG	28.3
SSNRA-MPS-NN	48.8
SSNRA-MPS-NC	35.2
SSNRSD-MPS-AVG	22.6
SSNRSD-MPS-NN	35.8
SSNRSD-MPS-NC	27.0
Dynamically reconfigured Dempster-Shafer combinations	
DS2TOP	59.8
DS3TOP	63.7
DS4TOP	65.7
DS5TOP	66.1
Dynamically reconfigured Mojority voting combinations	
MAJ3TOP	49.1
MAJ5TOP	54.7
Mojority voting of all other classifiers	
MAJORITY-ALL	65.5

and maintains updated success statistics for all classifiers. The predictions of the agent come from the current most successful classifier according to these statistics. The choice of the prediction strategy can have major impact on the MCML architecture, because different prediction approaches will eventually lead to different results. The following sub-section is therefore dedicated to the evaluation of the MCML architecture on different prediction methods and to further study the role of the meta-level in learning performance.

8.4.3 Evaluation of different MCML learning configurations

The objective of the experiments reported here is to evaluate three main aspects over the MCML approach:

1. Which prediction method works best?
2. What is the impact of metacognitive processing on learning performance?
3. What is the impact of the forgetting method in memory usage and learning performance?

Three methods for predicting the output category of the target object were evaluated:

1. Use the currently most successful classifier;
2. Use only the *DS5TOP* classifier, which was previously shown to be the most successful classifier; and
3. Use the majority voting of all classifiers.

The base classifier *SSNRA-MPS-NN* is used as baseline for comparison. For each learning configuration, five experiments per dataset were carried out. The results are summarized in Tables 8.3 and C.3.

Table 8.3: Summary of results obtained with different MCML configurations on LANGG68 and ALOI-1000 datasets

Dataset	Classifiers	Classifier used for prediction	Classifier ranking	Forgetting	Stopping condition	Avg. #cats learned	Avg. #iters	Avg. #inst/cat	Avg. protocol acc.(%)	Global acc.(%)
<u>LANGG68</u>										
	Only *SSNRA-MPS-NN*	*SSNRA-MPS-NN*	Not required	No	All cats. learned	68.0±0.00	1235.5±127.77	6.4±0.72	66.0±0.54	70.6±1.02
	All base classifiers	*DS5TOP*	Dynamic	No	All cats. learned	68.0±0.00	895.5±66.23	3.9±0.42	76.1±1.38	78.5±1.68
	All classifiers except *MAJORITY-ALL*	Current best	Dynamic	No	All cats. learned	68.0±0.00	877.8±62.49	3.9±0.34	74.9±3.58	77.9±1.09
	All classifiers	*MAJORITY-ALL*	Dynamic	No	All cats. learned	68.0±0.00	910.3±17.80	4.2±0.11	74.1±0.74	76.5±0.33
	All classifiers	Current best	Dynamic	Yes	All cats. learned	68.0±0.00	913.3±48.51	3.2±0.15	74.8±1.39	76.8±0.59
	All classifiers	Current best	Static	No	All cats. learned	68.0±0.00	850.5±64.57	3.7±0.43	77.0±3.34	78.8±1.93
	All classifiers	Current best	Dynamic	No	All cats. learned	68.0±0.00	855.6±77.25	3.8±0.41	77.0±2.56	78.1±1.37
<u>ALOI-1000</u>										
	Only *SSNRA-MPS-NN*	*SSNRA-MPS-NN*	Not required	No	Lack of data	88.2±19.83	3414.4±683.24	15.3±1.82	59.3±0.58	63.7±0.76
	All base classifiers	*DS5TOP*	Dynamic	No	Lack of data	225.4±26.24	6848.4±1007.99	10.8±0.88	63.5±0.74	67.6± 0.69
	All classifiers except *MAJORITY-ALL*	Current best	Dynamic	No	Lack of data	200.0±16.16	6023.6±611.91	10.8±0.34	67.2±3.22	67.6±0.34
	All classifiers	*MAJORITY-ALL*	Dynamic	No	Lack of data	231.4±30.00	6976.6±798.02	10.7±0.59	63.8±0.60	67.8±0.64
	All classifiers	Current best	Dynamic	Yes	Lack of data	140.6±37.04	4485.0±1284.02	8.1±0.55	62.0±0.89	66.1±0.39
	All classifiers	Current best	Static	No	Lack of data	199.2±42.50	5988.6±1299.84	10.8±0.42	62.7±0.48	67.4±0.54
	All classifiers	Current best	Dynamic	No	Lack of data	259.4±37.80	7789.0±1209.44	10.6±0.58	64.4±0.48	68.0±0.38

*Results in each row are summarized over 5 experiments.

In the experiments on LANGG68 dataset, the agent, for every MCML configuration, learned all 68 categories. When tested on the ALOI-1000 dataset, all experiments were concluded because of the "Lack of data" stopping condition. In the case of COIL-100 dataset, in a few experiments, all the categories were learned, whereas in several other experiments terminated when the images of a particular category ran out ("Lack of data"). For the ETH80 dataset, when using the baseline classifier setting, the agent reached the breakpoint in two experiments. In the rest of the experiments on ETH80, the agent learned all the categories.

With such results, a qualitative analysis of MCML configurations is possible only for the situations that are directly comparable. For example, for a given dataset, when all the experiments consistently concluded with the same stopping condition, results from such experiments can potentially be compared. This is the case with the experiments conducted on the LANGG68 and ALOI datasets. These results are depicted in the Table 8.3. On the other hand, for the COIL-100 and ETH80 datasets, the stopping condition is not consistent across all the experiments. A summary of the results on these two datasets are provided in Table C.3.

For the LANGG68 dataset, the MCML configuration using the *DS5TOP* prediction method led to better protocol and global accuracy when compared with the configuration using *MAJORITY-ALL* prediction method, as well as the configuration where *MAJORITY-ALL* classifier is not used and the prediction is made using the "current best" classifier. Since the later two configurations are working with multiple classifier combinations, in addition to the base classifiers, this result is quite interesting. This means, using less information and fewer computations, *DS5TOP*, which only uses the base classifiers, is able to achieve better performance than the other two configurations, which are comparably more elaborate and computationally expensive.

However, the conclusions on these three configurations can not be generalized across other datasets. For example, in the case of the ETH80 dataset, all 8 categories were learned using these configurations, the *MAJORITY-ALL* configuration led to the best performance. In this case, extra set of classifier combinations do lead to better performance than just using the base classifiers.

Such analysis is difficult to be carried out on the COIL-100 and ALOI-1000 datasets, since most of the experiments terminated due to the "Lack of data". However, for the three configurations, the average number of learned categories were similar, repesectively for both the datasets, certain general conclustions can be made. Similar to the LANGG68 dataset, for COIL-100, *DS5TOP* gave the best accuracy values, whereas for ALOI-1000, the best performance was achieved using the "current best" prediction.

Amongst the three configurations being discussed, no single configuration can be considered better than others. Depending on the dataset (*i.e.* the type of data), different confgurations performed differently. Having mentioned this, the difference in performance, in terms of the accuracy measures, the memory storage (the number of instances stored per category) or the training time (the number of iterations) is not remarkable. In fact, the performance of these configurations is not very different from the original MCML configuration.

To evaluate the impact of meta-cognitive processing in the number of learned categories, a comparison was done with the best base classifier, namely, *SSNRA-MPS-NN*. On all evaluation metrics, this classifier performs poorer than any other configuration.

The forgetting method (described in Section 5.5) was evaluated by comparison with the original *MCML* approach. For the two datasets, LANGG68 and ETH80, where both approaches learned all the categories, the forgetting method enabled the reduction of memory consumption in 15% and 31% respectively, at the expense of spending 6% more time in learning the categories. Similar analysis cannot be derived for the other two datasets, because the protocol halted due to lack of images in all experiments, and the number of learned categories vary greatly from one configuration to another to reach any legible conclusions.

Finally, the original *MCML* approach, where classifier combinations are based on dynamic ranking of base calssifiers, is compared with the configuration where this ranking is static. The base classifiers in the new configuration are ranked with respect to their performance in offline evaluations (as obtained in Section 7.3). For the LANGG68 dataset, all the evaluation metrics lead to very similar results, whereas for the ETH80 dataset, dynamic ranking leads to slightly better accuracy at the cost of number of test iterations and memory storage. In the two experiments, using these configurations, all COIL-100 categories were learned. In this case also, the performance difference between the configurations is minor. The static ranking led to better accuracy at the cost of memory storage and the number of test iterations. For the rest of the experiments with COIL-100 and all the experiments with ALOI-1000, similar analysis cannot be carried out due to the premature halt of the protocol and the large difference in the number of learned categories, using these configurations.

8.5 Grounding spoken words

The category learning architecture designed for grounding spoken vocabulary (described in Section 3.4) is supported by the learning approach detailed in Chapter 6. For the experiments conducted, the classifiers for predicting word categories are based on *Phoneme-MFCC similarity measure*[3]. This section is dedicated to the evaluation of this learning approach at the task of open-ended word learning and category acquisition.

Unlike the evaluations of the other learning architectures, where experiments were conducted over various object images datasets and the role of teacher was taken by a simulated

[3]The *Edit distance* based similarity measure was introduced only after the experiments described in this section were completed, therefore the evaluation presented in here does not include classifier based on *Edit distance*, presented in Section 6.2.2. Although, an independent comparison of *Phoneme-MFCC* and *Edit distance* based classifiers was conducted, as presented previously in Section 7.7

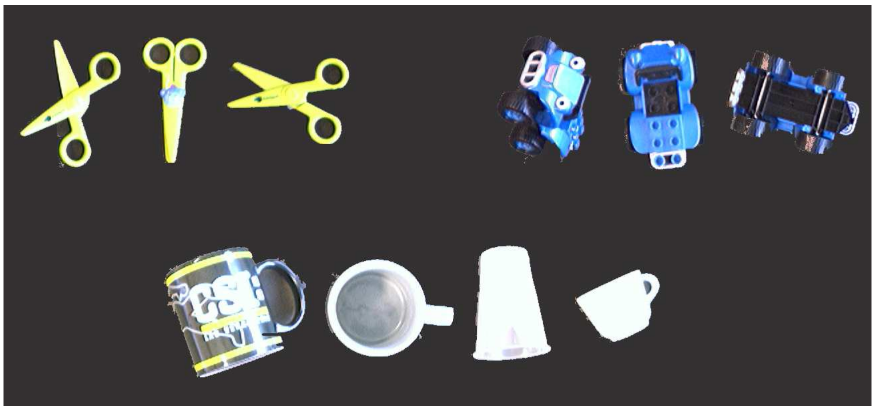


Figure 8.7: A set of examples illustrating different views in which objects were shown to the robot: the instance of *scissor* in varying degrees of blade openings; the instance of *jeep* in different 3D orientations (front towards the camera, top view and bottom facing the camera); and different instances from the *cup* category in varying orientations.

---

teacher, here a human user follows the teaching protocol and teaches names of objects present in a scene shared with the robot. As mentioned earlier in Section 8.1.2, in this case, f in Eqn 8.1 is set to 0, which makes the protocol function similarly to the original protocol.

Although the system is designed to be open-ended, for the reported experiments, the number of categories taught to the robot was set to 13 (8 toys, 3 regular cutlery and 2 office objects). Two experiments were carried out in which categories were introduced in different sequences. The robot began with zero knowledge and gradually built its vocabulary. In the first experiment, categories were introduced in the following sequence:

cup – 4 different objects in this category
star – a star shaped toy
jeep – a toy jeep
scissor – a toy scissor
car – a toy car
horse – a toy horse
fork – 2 different objects
stapler – 2 different objects
knife – 1 object
train – a toy train
bike – a toy bike
boy – a toy in the shape of a small boy
screwdriver – 2 different objects

For the second experiment, categories were introduced in the following sequence:

cup, star, jeep, horse, car, bike, fork, knife, stapler, screwdriver, scissor, boy, train.

It is important to note that there was no control over how the user chose to show objects to the robot. That is, unlike LANGG68 dataset (which was developed by us), each object category consists of highly heterogeneous set of object instances. presented by a To give the reader an idea of how various objects were shown, 8.7 provides an illustration. The visual features from the same object can vary significantly depending on how the user decides to demonstrate that object to the robot. In a similar manner, different objects belonging to the same category also lead to very different visual features. However, the learning model of the robot is designed such that the spoken words guide object category formation. The varying visual features will aggregate together to form a single object category description if the agent identifies that they share the same name (word category).

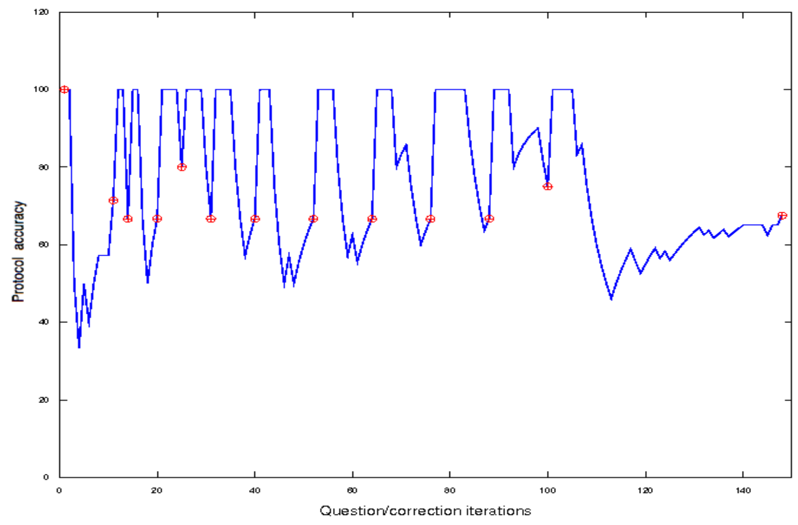
8.5.1 Global evaluation

The agent was able to learn the 13 names and respective meanings in both experiments. Fig. 8.8 presents the evolution of protocol accuracy and Table 8.4 provides a summary of these experiments. Essentially, the learning behavior is similar to that observed for the rest of the architectures.

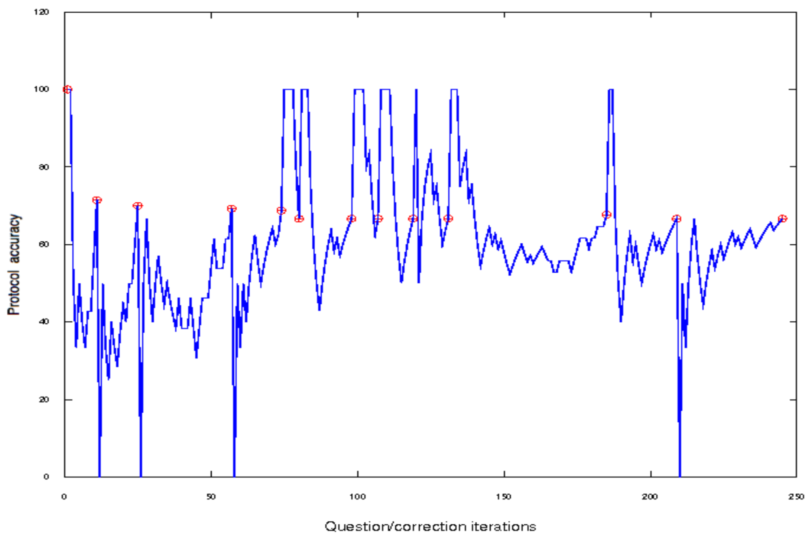
Although in both experiments the robot was able to ground the word categories in their respective object categories, as the number of categories taught to the robot increased, it became harder for the agent to successfully associate word categories to object categories. This can be noticed for both experiments where towards the end of their respective graphs, the interval between the introduction of new categories increases. The length of such interval is an indicator of the amount of effort spent by the agent in reorganizing its categories.

Comparing the two experiments, it is visible that in the second experiment the agent spent much more time in the recovery process. The number of question/correction iterations required to learn 13 categories in the second experiment was 245, which is almost twice as that for the first experiment. In the second experiment, after the introduction of the second category, the agent begins to show difficulty in forming correct category descriptions. The same pattern can be noticed after the introduction of the third, fourth, twelfth and thirteenth category.

Since the global accuracy and the average protocol accuracy measures are computed as an average over all the iterations, they reflect the learning performance over a complete experiment. Since the agent had more difficulties in learning categories in the second experiment, the values of these measures for the second experiment are lower. On the other hand, for the first experiment, over all the iterations, protocol accuracy remained mostly above the accuracy threshold. Therefore, both evaluation measures are higher for the first experiment than those for the second experiment (see Table 8.4).



(a)



(b)

Figure 8.8: Evolution of protocol accuracy versus number of question/correction iterations for a accuracy threshold of 66.67%: (a) experiment 1; and (b) experiment 2.

The main reason behind the difference in the performances of these experiments is linked to the incremental nature of the learning algorithm. For incremental machine learning algorithms, the order in which new data gets introduced has a great impact on the evolution of the learning performance. For both experiments, there were no constraints over how the user introduced new visual and vocal signals to the agent. The only external control was the order in which new categories can be introduced and the rules of the teaching protocol. We

Table 8.4: Summary of experiments

Exp#	Accuracy threshold	#iterations	#presented categories	Global accuracy (%)	Avg. protocol accuracy (%)
1	66.67	148	13	68.24	75.35
2	66.67	245	13	63.27	60.14

believe, the sequence in which the learning agent received the visual and vocal signals in the first experiment helped the agent in making better word and object category descriptions, early on. However, it is evident from both experiments that the agent was able to learn incrementally and reach the accuracy threshold for each of the introduced categories.

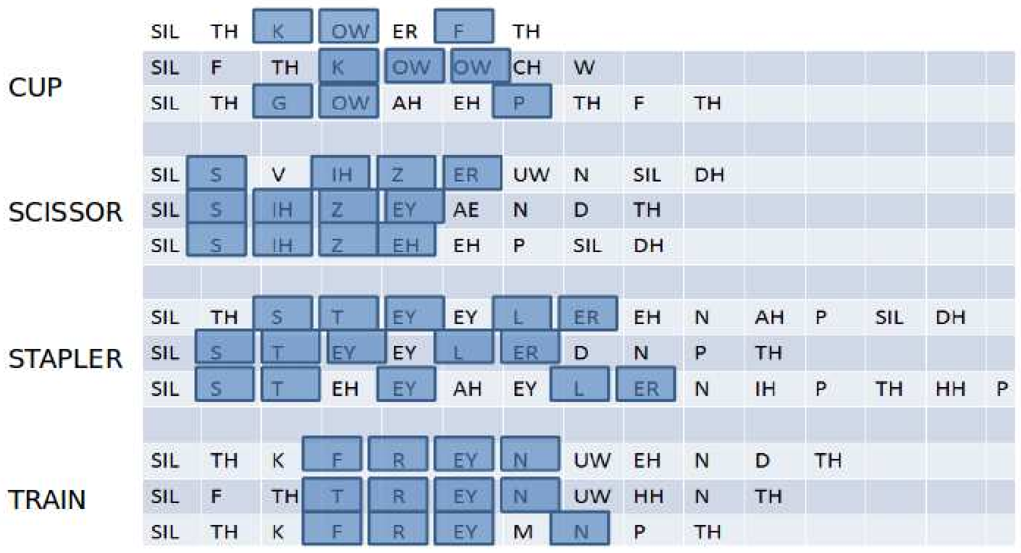
Overall, global accuracy in both experiments is fairly low. This implies that, although the agent shows learning, in the current state its discrimination capability is limited. The main reason behind the poor performance is the uncertainty in the category label. In the present scenario, the use of spoken words adds extra complexity to the vocabulary grounding problem. As a perceptual input, a spoken word (audio signal) is uncertain. Classifying a spoken word as belonging to a word category relies on the *Phoneme-MFCC* measure of similarity presented in Section 6.2.2. A key component in computing this measure is the predicted phoneme sequence for a spoken utterance. However, the phoneme sequences predicted for the spoken utterances of the same word can vary a lot (see Figure 8.9). Such predictions add a significant amount of noise to a word representation and, more generally, to word categories.

8.6 Summary

This chapter focused on the evaluation of several open-ended category learning approaches proposed in this thesis. It was argued that the evaluation of open-ended learning approaches cannot be carried out using traditional methods. In our previous work (Seabra Lopes and Chauhan, 2007), a *teaching protocol* was proposed, which was designed specifically to evaluate the ability of a learning agent to scale up to larger sets of categories. However, in its original form, this protocol has several drawbacks concerned mostly with its exhaustive nature. An enhanced version of this protocol was proposed which remedies these drawbacks. Evaluations of all the learning approaches was then carried out using the new protocol.

A set of experiments was carried out to compare instantiations of the three architectures designed for grounding textual vocabulary.

The different approaches were evaluated, using a simulated teacher, on four object-images datasets of varying complexity. The multi-classifier approach (MCML) led to the most promising results (learning approximately 300 categories). The SVDD-based approach was found to



CUP	SIL	TH	**K**	**OW**	ER	**F**	TH								
	SIL	F	TH	**K**	**OW**	**OW**	CH	W							
	SIL	TH	**G**	**OW**	AH	EH	**P**	TH	F	TH					
SCISSOR	SIL	**S**	V	**IH**	**Z**	**ER**	UW	N	SIL	DH					
	SIL	**S**	**IH**	**Z**	**EY**	AE	N	D	TH						
	SIL	**S**	**IH**	**Z**	**EH**	EH	P	SIL	DH						
STAPLER	SIL	TH	**S**	**T**	**EY**	EY	**L**	**ER**	EH	N	AH	P	SIL	DH	
	SIL	**S**	**T**	**EY**	EY	**L**	**ER**	D	N	P	TH				
	SIL	**S**	**T**	EH	**EY**	AH	EY	**L**	**ER**	N	IH	P	TH	HH	P
TRAIN	SIL	TH	K	**F**	**R**	**EY**	**N**	UW	EH	N	D	TH			
	SIL	F	TH	**T**	**R**	**EY**	**N**	UW	HH	N	TH				
	SIL	TH	K	**F**	**R**	**EY**	M	**N**	P	TH					

Figure 8.9: Phoneme sequences predicted in 3 separate utterances of words cup, scissor, stapler and train, respectively. Highlighted phoneme sequence for each utterance represents the ideal prediction. The non-highlighted phonemes constitute noise in a predicted sequence.

be more suitable than the component-based approach for learning the categories in the most homogeneous datasets. The component-based approach was better than SVDD for the rest of the datasets. For all the four datasets, the multi-classifier approach consistently learned greater number of categories, with least memory usage.

A detailed analysis of the multi-classifier approach was carried out for the longest experiment. Predominantly, dynamically reconfigured classifiers based on majority vote and Dempster-Shafer theory of evidence, performed better than the base classifiers (DS5TOP approach provided best overall performance). Similar results were observed in other experiments with MCML.

An additional set of experiments was carried out to evaluate different MCML configurations. In the experiments, MAJORITY-ALL and DS5TOP classifiers outperform other prediction methods. Overall, performances of different MCML configurations are very similar. Forgetting led to a noticeable drop in memory usage at at a relatively small drop in of classification accuracy.

Finally, the approach to grounding spoken words (GSW) was evaluated at grounding 13 words. A human user showed objects and taught names of these objects to the agent. In the reported experiments, the agent was shown to successfully learn the object categories and the corresponding word categories (object names).

9 Conclusions and discussion

This thesis addresses the problem of word learning for human-robot interaction. In order to interact and communicate with the human users using natural language, adapting to different users, tasks and environments, robots need to acquire language from the users themselves. A social language grounding scenario was designed, where a human or a simulated instructor taught the agent the names of the objects present in their shared visual environment. The present work was carried out with a particular concern for the fact that word learning is an open-ended domain. Several learning architectures were explored that can be used by robotic agents for long-term and open-ended category learning. The problem of scaling-up to larger vocabularies (and hence categories) was addressed using these architectures and the corresponding learning approaches. The work carried out in this thesis led to the following key contributions:

– A robotic agent was designed which acquires vocabulary through a teacher (human or another agent), and grounds this vocabulary in visual perception. Designing such agent required integration of techniques from the areas of human-robot interaction, computer vision, machine learning and speech processing. The agent successfully demonstrated the capability to ground object names and learn object categories in an incremental and open-ended manner in diverse experimental settings.

– Multiple object representations were explored with focus on capturing shape information, namely several shape signatures and a graph-based shape representation. Several of these representations are original proposals. Three general measures of similarity

(Euclidean Similarity, Pyramid Match Score and Manhattan-Pyramid Similarity) were used for comparing the shape signatures. Manhattan-Pyramid Similarity measure is an original contribution, which was designed to resolve an anomaly in "pyramid matching by minimization". A novel similarity measure for comparing the graph-based representation was also presented.

- The representations and similarity measures were used to develop several computationally light instance-based and cluster-based classifiers. These classifiers involve a combination of a representation, a similarity measure and a decision rule (nearest-neighbor, nearest-cluster and average classification rules were used). An independent evaluation of these classifiers allowed an assessment of different similarity measures, shape signatures and decision rules. With respect to the similarity measures, classifier configurations using Manhattan-Pyramid Similarity consistently led to better performance. From different shape signatures, *Shape Slices Histogram* and *Shape Slices Normalized Radii Averages* performed the best. Amongst the decision rules, the nearest-neighbor rule outperformed the other two.

- Four learning architectures were proposed that can be used by the robotic agent for online, long-term and open-ended category learning. Instantiations of these architectures were implemented and evaluated at the task of open-ended category learning. The following architectures and the corresponding instantiations were designed:

  - Architecture based on instance-based learning (Section 3.3.1):

    [**COMP**] - The categories are represented by the graphs of components of individual instances, and classification is achieved by finding graph similarities.

  - Architecture based on one-class learning (Section 3.3.2):

    [**SVDD**] - The learning and classification approach is based on genetic SVDD (described in Section 5.3.3). The original implementation of SVDD was modified and extended to support open-ended learning. The genetic approach to SVDD parameter optimization, for a multi-class learning scenario, is an original contribution of the thesis.

  - Architecture based on multiple classifiers with meta-learning (Section 3.3.3):

    [**MCML**] - The learning and classification is based on using multiple base-classifiers, classifier combinations and a metacognitive component which maintains updated success statistics for all the classifiers for dynamically reconfiguring the classifier combinations (see Section 5.4).

  - Architecture for grounding spoken words (Section 3.4):

    [**GSW**] - Unlike the other three architectures, where category names are input as strings, here words are communicated via speech. Spoken word categories are learned using unsupervised clustering which leads to dynamic formation and reorganization of object categories. Classification of new instances is supported by multiple classifiers and combinations and a metacognitive component (as in MCML).

– Due to the lack of suitable approaches to evaluate open-ended learning algorithms, in our previous work (Seabra Lopes and Chauhan, 2007), a teaching protocol was proposed. An improved version of this protocol is proposed in this thesis (Section 8.1.2). This protocol can be useful for comparing the word learning capabilities of different agents as well to assess research progress with respect to scaling-up to larger vocabularies. The new protocol is used for evaluating the instantiations of the learning architectures, as well as several MCML configurations.

The evaluation of the agent at the task of open-ended learning showed that the number of learned categories varied greatly, depending on the learning approach. The experimental evaluations were carried out on several object-images datasets. The learning approach is the main factor in the speed of evolution, recovery and eventual breakpoint. Although each architecture was successful at learning in an open-ended manner, their respective success at scaling-up to larger vocabularies was limited by the learning approach used. Two specific learning behaviors were observed: either the agent continues to acquire new words and categories, or the agent is no longer able to accommodate new categories (*the breakpoint is reached*). In several cases, the evaluation halts due to lack of data in the dataset.

These results raise several issues for discussion. One of the main issues is concerned with the existence of a breakpoint in the learning capacity of a robot. The existence itself seems easy to accept. Robots are limited in their perceptual (sensors, sensor fusion, active sensing) and sensori-motor abilities and, therefore, cannot learn arbitrarily large numbers of categories, particularly when perception and action do not enable them to detect small between-category differences.

For SVDD and COMP, breakpoint was reached in several experiments. As the number of words increases, the training becomes more difficult and some class descriptions have to be corrected many times before the accuracy threshold is achieved. For these two architectures, this is true even in the cases where breakpoint was not reached. Eventually, on many experiments, the learning capacity of the agent, on these two architectures, reaches its breakpoint.

Number of categories learned over SVDD and COMP approaches ranged from 3 to 37. This is more problematic. A robot with such a limitation will not be of any use in environments that require language-based interaction with users. Why can't these systems learn more words and categories? In particular, why can't they learn more concrete object categories and names? These are actually very general questions. Any approach which is incapable of scaling-up to greater number of categories is of no use in real-world scenarios. Multiple factors can limit the scaling-up:

– sensor limitations

– lack of active sensing/animate vision

– lack of physical interaction with the target objects

– lack of consideration of the affordances of objects

– limited interaction between the learning agent and the human caregiver

– inappropriate perceptual abilities and representations

– inappropriate category learning methods.

With respect to sensing, our agent is very limited. A single camera enables perception of target objects, and it remains in a fixed position over the objects. There is no possibility either for active/animate sensing or for sensorimotor experience with the objects. This limitation is common to many of the prototypes and models described in literature.

The focus of most of works on word-grounding has been on grounding vocabulary predominantly in visual perception. Such systems are typically unable to take into account the affordances of objects, that is, the actions and uses that they afford. Interestingly, an early explanation for the shape bias, observed in children when learning concrete object names, was based on the conjecture that shape would be a strong determiner of affordances (Rosch, 1973). Gibson (1979) also stressed the importance of affordances in visual perception. An early robotic model that classifies affordances (the produced categories are here called proto-symbols) and uses them to guide navigation was described by MacDorman et al. (2000). In recent years, some research groups focus explicitly on designing affordance based vocabulary grounding models (Iwahashi, 2006; Iwahashi et al., 2010; Krunic et al., 2009; Sugiura and Iwahashi, 2007; Takamuku et al., 2006).

Our approach explicitly includes the human user as instructor or mediator for the word learning process. However, the initiative for interaction between the instructor and the agent is always on the instructor's side. One possible future direction can be to create richer human-agent interaction scenarios where the agent may also wish to ask for instructor's help in the language acquisition process. This results in a mixed-initiative interaction that allows for dual control and mutual gearing, as observed in the relation between infant and caregiver (Cowley, 2006, 2007).

The limitations discussed so far are concerned with the "external" component of language acquisition and symbol grounding. Meanwhile, the internal mechanisms should also be significantly improved. The learning approaches that have been used until now in most systems are primarily connectionist, probabilistic or instance-based. In addition, in this thesis we proposed architectures based on: SVDD (one-class learning with support-vector data descriptions); and MCML (multiple classifiers with meta-learning). Overall, MCML was shown to be the most successful amongst all the tested learning methods. In the longest experiment, 293 categories were learned using MCML. Additionally, in all of the experiments using this approach, breakpoint was not reached and, at the moment experiments stopped, the rate of learning showed no significant decline. This leads to the conclusion that using MCML, many more categories could have been learned, if not for the limitation imposed by the experimental settings. Although different published works on word learning are not directly comparable to each other or to the above presented work (due to their many specificities), no other approach in literature, in our knowledge, has reported grounding of this many words and open-ended learning of this many categories.

Although the work in this thesis focused on visual category learning, it would be interesting and relevant to extend the proposed learning architecture to other domains, including sensori-motor categories. Sensori-motor categories can be learned through action, either self-performed or observed. Apart from action categories themselves, this is relevant for taking affordances into account when categorizing objects.

The "symbol grounding problem" (Harnad, 1990) was proposed as a critique of classical symbolic AI. Purely symbolic systems, which were theorized as powerful models of cognition (Newell and Simon, 1972; Simon, 1979), do not account for the meanings of symbols. Computational processes that function exclusively on symbols cannot be considered cognitively plausible (Barsalou, 1999; Harnad, 1990; Pinker, 1984). Agents (robotic or otherwise) with appropriate embodiments, interaction and learning capabilities can ground symbols in their sensori-motor perception. As the field of language grounding has grown over the years, it is becoming an exciting prospect that some of the future directions will definitely involve concomitant use of grounded symbols with computational processes that operate over symbols (as was the case in classical AI). Some of the recent works in literature have started to combine symbolic reasoning with grounded words (see for example Connell et al., 2012). This is also one of the possible directions of continuation of the work in thesis.

As a final note, as can be seen from this discussion, there is plenty of work for the robotics, AI and cognitive science communities concerning the development of artificial agents able to acquire extended vocabularies. Designing agents that can reach human-level vocabulary should remain one of the key research focus and a long-term goal of these communities. However, there are preliminary goals to be reached and perhaps we should begin by aiming towards chasing Chaser (Pilley and Reid, 2010), as far as word learning capabilities are concerned.

A
Robotic-arm control

To pick an object using the SG6-UT robotic arm (see Figure A.1), it is necessary to know the exact placement and the orientation of that object. Since the camera is the only available sensor, the suitable picking position is found by further processing the extracted object image. A linear mapping between camera and arm coordinates is assumed.

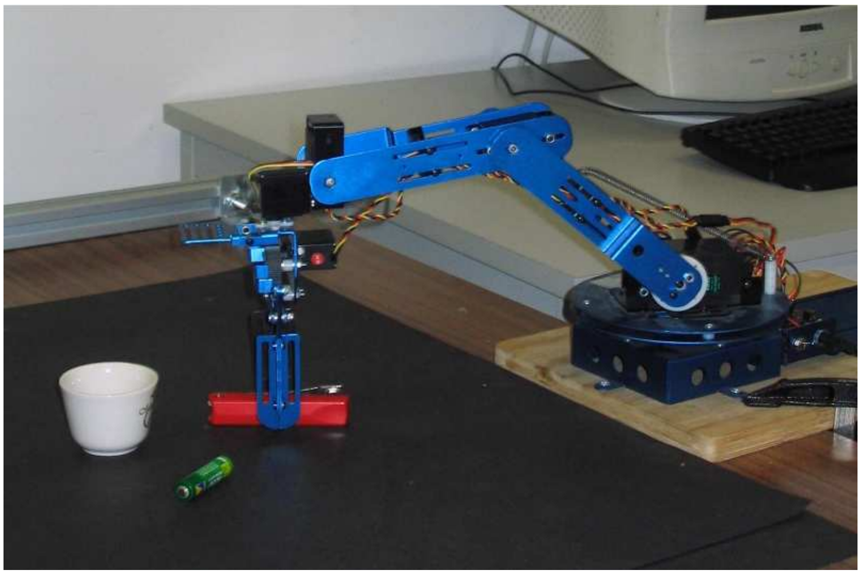


Figure A.1: The CrustCrawler SG6-UT robotic arm – put more images

Finding grasp position and orientation

The object should be grasped such that it does not rotate or fall down. A suitable picking location (see Fig. A.2) is found by taking the following conditions into account:

1. the gripper should be centered at the geometric center of the object;
2. the distances A_1B_1, A_2B_2 and A_3B_3 must be less than the maximum gripper width;
3. the areas of the triangles formed by the points $A_1A_2A_3$ and $B_1B_2B_3$ must be less than a threshold (ideally zero); and
4. angles α and β between the gripper ends and lines A_1A_3 and B_1B_3 respectively, should be close to zero.

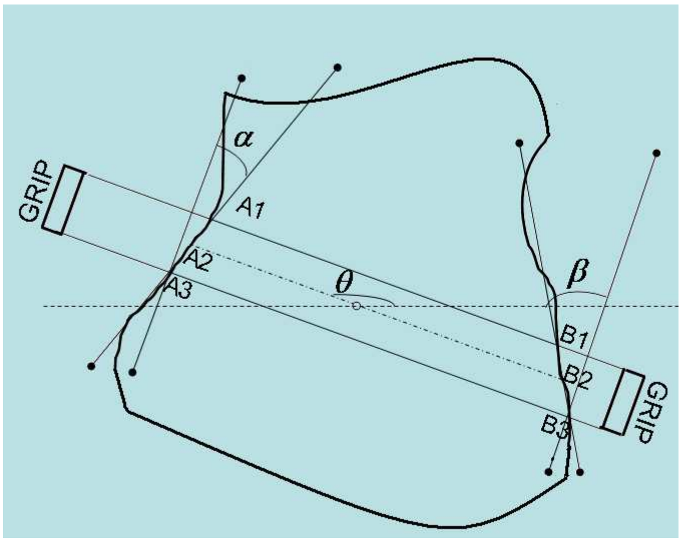


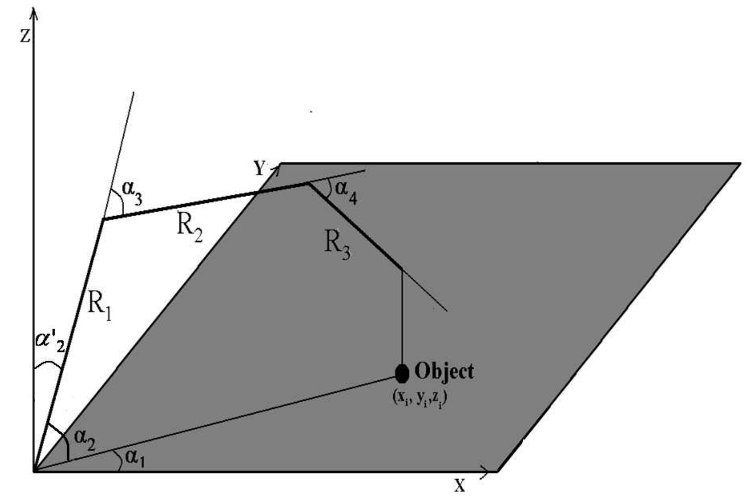
Figure A.2: A hypothetical object and one possible gripper position.

Depending on the object's structure, there will be more than one possible location to pick it up. The first suitable location to pick the object is used. Once the joint angles have been obtained, these values are passed to the arm's servo controller, for it to pick up the object.

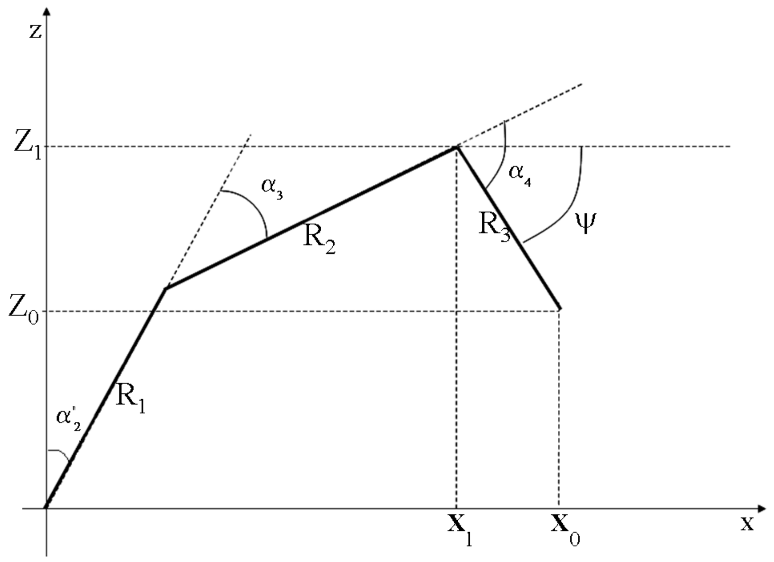
Arm kinematics

Figure A.3a shows a simplified model of the arm. R_1, R_2 and R_3 are the arm link lengths and α_1, α_2', α_2, α_3 and α_4 are the joint angles to be determined. α_1 can be easily found by using basic trigonometry:

$$\alpha_1 = \tan^{-1}\left(\frac{y}{x}\right) \tag{A.1}$$



(a)



(b)

Figure A.3: (a) A simplified model showing arm joints and the object location; (b) Robotic arm plane

To calculate the rest of the angles, consider the plane of the robotic arm, as in Fig. A.3b.

x_0 and z_0 are given by:

$$x_0 = R_1 \sin(\alpha_2') + R_2 \sin(\alpha_2' + \alpha_3) + R_3 \sin(\alpha_2' + \alpha_3 + \alpha_4) \tag{A.2}$$

$$z_0 = R_1 \cos(\alpha_2') + R_2 \cos(\alpha_2' + \alpha_3) + R_3 \cos(\alpha_2' + \alpha_3 + \alpha_4) \tag{A.3}$$

We have two equations and three variables to determine. To solve this problem, the angle ψ between the third link and the horizontal coordinate is forced to be $\pi/2$. This means, the wrist of the robotic arm will always be perpendicular to the base of the table. This causes no significant problem, since it is the best position for a successful grasp. Given this, Z_1 and X_1 can be determined by:

$$Z_1 = Z_0 + R_3 \tag{A.4}$$

$$X_1 = X_0 \tag{A.5}$$

Using equations A.2, A.3, A.4 and A.5, α_3 and α_2' can be determined as:

$$\boxed{\alpha_3 = \pm \cos^{-1}\left[\frac{Z_1^2 + X_1^2 - R_1^2 - R_2^2}{2R_1R_2}\right] \qquad \text{(A.6)}}$$

$$\boxed{\alpha_2' = \pm \tan^{-1}\left[\frac{X_1(R_1 + R_2\cos(\alpha_3)) - Z_1R_2\sin(\alpha_3)}{Z_1(R_1 + R_2\cos(\alpha_3)) - Z_1R_2\sin(\alpha_3)}\right] \qquad \text{(A.7)}}$$

Now the final unknown angle can be given as:

$$\boxed{\alpha_4 = \psi - \alpha_2' - \alpha_3 \qquad \text{(A.8)}}$$

B
Accuracy in multi-class learning scenario

It is generally assumed that many metrics developed for binary classification can be directly extended to evaluate the performance of multi-class learning algorithms (Holt et al., 2010; Sokolova and Lapalme, 2009). However, this is not as straight forward as has been suggested in literature and for the same measures multiple interpretations exist. For example, metrics for evaluating multi-class algorithms outlined in surveys by Sokolova and Lapalme (2009) and Holt et al. (2010) were derived by generalizing the existing binary classification metrics. Although, in both the surveys, the derived generalized metrics are very different form each other. Since *accuracy* is the metric of choice for this thesis, in this appendix we will take a look at two common generalizations of this metric.

Given a multi-class scenario, where the learning algorithm was trained for N different classes, the evaluation involves classification of independently drawn and previously unseen test instances from these classes. Each test instance gets assigned to one of the N classes leading to a multi-class $N \times N$ confusion matrix, M (see Table B.1).

		Predicted class			
		a	b	c	d
True class	a	10	2	3	0
	b	1	25	2	5
	c	4	2	15	7
	d	2	0	1	30

Table B.1: A hypothetical example of a 4-class confusion matrix

According to (Holt et al., 2010),*accuracy* in multi-class scenario is given as:

$$Acc_1 = \frac{1}{|M|}\sum_{i=1}^{N} M_{i,i} \tag{B.1}$$

where $|M| = \sum_{i,j=1}^{N} M_{i,j}$. That is, *accuracy* is defined as the ratio of all the correctly classified test instances (diagonal elements of the confusion matrix) to the total number of instances used for testing. In case of binary classification (see Table B.2), B.1 will be reduced to :

$$Acc_1 = \frac{tp + tn}{tp + tn + fp + fn} \tag{B.2}$$

which is the standard definition of *accuracy* in the binary-class scenario.

		Predicted class	
		pos	neg
True class	pos	true positive (tp)	false negative (fn)
	neg	false positive (fp)	true negative (tn)

Table B.2: Confusion matrix for binary classification

According to (Sokolova and Lapalme, 2009), *accuracy* in multi-class scenario is given as:

$$Acc_2 = \frac{1}{N}\sum_{i=1}^{N} \frac{tp_i + tn_i}{tp_i + tn_i + fp_i + fn_i} \tag{B.3}$$

That is, *accuracy* is defined as the average of *accuracy* values computed separately for all N classes. In a binary-class scenario, B.3 reduces to:

$$Acc_2 = \frac{tp + tn}{tp + tn + fp + fn} \tag{B.4}$$

As can be noticed, equations B.2 and B.4 are same in case of binary classification. However, this is not the case for multi-class problems. For the example in Table B.1, using equation B.1 we get $Acc_1 = 0.74$ and using equation B.3 we get $Acc_2 = 0.87$ (see Fig.B.1(a)).

It is evident here that, depending on how *accuracy* is generalized from a binary-class to a multi-class scenario, different interpretations led to very different conclusions regarding the performance of the multi-class learning algorithms.

10	2	3	0
1	25	2	5
4	2	15	7
2	0	1	30

10	5
7	87

(a)

$acc_1 = 0.89$

10	2	3	0
1	25	2	5
4	2	15	7
2	0	1	30

25	8
4	72

(b)

$acc_2 = 0.89$

10	2	3	0
1	25	2	5
4	2	15	7
2	0	1	30

15	13
6	75

(c)

$acc_3 = 0.83$

10	2	3	0
1	25	2	5
4	2	15	7
2	0	1	30

30	3
12	64

(d)

$acc_4 = 0.86$

$Acc_2 = 0.87$

Figure B.1: Steps in computation of Acc_2 for the example in Table B.1

To explore the reason behind the difference between the Holt's and Sokolova's interpretations of *accuracy*, let's begin by rewriting equation B.3 in terms of a multi-class confusion matrix as:

$$\begin{aligned} Acc_2 &= \frac{1}{N}\left(\sum_{i=1}^{N} \frac{M_{ii} + \sum_{j,k\neq i} M_{jk}}{|M|}\right) \\ &= \frac{1}{N|M|}\left(\sum_{i=1}^{N} M_{ii}\right) + \frac{1}{N|M|}\left(\sum_{i=1}^{N}\sum_{j,k\neq i} M_{jk}\right) \end{aligned} \tag{B.5}$$

Using B.1, Acc_2 can be written as:

$$Acc_2 = \frac{Acc_1}{N} + B \tag{B.6}$$

where

$$B = \frac{1}{N|M|}\left(\sum_{i=1}^{N}\sum_{j,k\neq i} M_{jk}\right) \tag{B.7}$$

Equation B.7 can be expressed such that a clear distinction between the contribution of diagonal elements of the confusion matrix (correctly classified instances) and the off-diagonal elements (incorrectly classified instances) can be made (see eq. B.8).

$$\begin{aligned} B &= \frac{1}{N|M|}\sum_{i=1}^{N}\left(\underbrace{\sum_{j\neq i} M_{jj}}_{\text{diagonal elements}} + \underbrace{\sum_{\substack{j,k\neq i\\ j\neq k}} M_{jk}}_{\text{off-diagonal elements}}\right) \\ &= \frac{1}{N|M|}\left(\sum_{i=1}^{N}\sum_{j\neq i} M_{jj}\right) + e \end{aligned} \tag{B.8}$$

where

$$e = \frac{1}{N|M|}\left(\sum_{i=1}^{N}\sum_{\substack{j,k\neq i\\ j\neq k}} M_{jk}\right) \tag{B.9}$$

with limits $0 \leq e < 1$

The expression e (eq. B.9) is the amount of explicit contribution by the off-diagonal elements (*i.e.* incorrect predictions) of the confusion matrix to Acc_2. Further expanding eq. B.8, we get:

$$B = \frac{1}{N|M|}\left((N-1)\sum_{i=1}^{N} M_{ii}\right) + e \tag{B.10}$$

Using eq. B.1, we can modify the above expression to:

$$B = Acc_1 - \frac{Acc_1}{N} + e \tag{B.11}$$

By substituting B from equation B.11 to eq. B.6, we get:

$$Acc_2 = \frac{Acc_1}{N} + \left(Acc_1 - \frac{Acc_1}{N} + e\right) \tag{B.12}$$

$$\boxed{Acc_2 = Acc_1 + e \qquad \text{(B.13)}}$$

Equation B.13 shows that the averaged *accuracy* as mentioned in Sokolova and Lapalme (2009) is a summation of *accuracy* (as mentioned in Holt et al. (2010)) and "e". As mentioned earlier, the term "e" (eq. B.9) is computed using only the off-diagonal elements (*i.e.* incorrect predictions). That is, Acc_2 is overestimating the the accuracy of a multi-class learning algorithm by "e". Whereas, this is not the case for Acc_1 (eq. B.1) which is computed as a ratio of all correct predictions to the total number of predictions for a given set of test instances. Therefore, Acc_1 is a more reliable estimate of the *accuracy* of multi-class learning algorithm.

C
Evaluation results

Table C.1: Summary of 10-fold cross-validation evaluation on LANGG68, COIL-100, ETH80 and ALOI-1000 datasets trained using one-step learning

Feature space	Similarity measure	Classifier type	Avg. acc. ± stdev (%)			
			LANGG68	COIL-100	ETH80	ALOI-1000
AREA	*ES/MPS*	*AVG*	30.8±1.11	11.1±0.84	42.5±1.42	**22.6±0.38**
AREA	$P_\triangle$	*AVG*	**33.5±0.96**	**11.8±0.60**	**44.8±2.29**	5.0±0.10
AREA	*ES/MPS*	*NC*	28.53±1.16	8.68±0.85	38.38±2.24	**21.3±0.31**
AREA	$P_\triangle$	*NC*	**31.78±1.19**	**12.15±1.34**	**42.32±2.87**	13.3±0.31
AREA	*ES/MPS*	*NN*	**27.6±1.56**	7.4±0.93	**37.5±1.92**	15.0±0.40
AREA	$P_\triangle$	*NN*	27.4±1.26	**7.6±0.93**	37.2±1.60	**15.11±0.30**
RADSD	*ES/MPS*	*AVG*	21.0±0.65	6.5±0.78	34.6±2.25	**2.1±0.16**
RADSD	$P_\triangle$	*AVG*	**23.6±0.83**	**7.9±0.64**	**38.4±2.67**	1.7±0.07
RADSD	*ES/MPS*	*NC*	18.57±0.98	5.63±0.87	30.6±1.2	**1.6±0.09**
RADSD	$P_\triangle$	*NC*	**20.46±1.29**	**6.89±0.94**	**33.1±2.3**	1.4±0.10
RADSD	*ES/MPS*	*NN*	**17.1±0.91**	5.0±0.64	**29.2±2.31**	2.0±0.11
RADSD	$P_\triangle$	*NN*	**17.1±1.34**	**5.2±0.45**	29.1±1.99	2.0±1.14
SLH	*ES*	*AVG*	48.2±1.43	32.4±0.95	41.3±0.66	21.2±0.27
SLH	*MPS*	*AVG*	**54.2±1.17**	**38.4±1.34**	**49.1±2.12**	**22.9±0.34**
SLH	$P_\triangle$	*AVG*	46.0±1.77	27.7±0.76	41.5±3.34	12.1±0.22
SLH	*ES*	*NC*	62.35±0.84	55.71±1.96	59.9±2.1	43.8±0.27
SLH	*MPS*	*NC*	**66.64±1.25**	**61.96±0.98**	**63.72±2.89**	**45.5±0.44**
SLH	$P_\triangle$	*NC*	64.88±1.02	58.59±2.02	62.11±3.16	42.8±0.39
SLH	*ES*	*NN*	69.9±1.76	65.8±1.15	64.7±3.73	52.7±0.43
SLH	*MPS*	*NN*	72.0±1.30	**70.2±1.91**	**67.6±1.38**	**54.1±0.52**
SLH	$P_\triangle$	*NN*	**72.4±1.46**	69.9±2.03	67.1±3.40	**54.1±0.42**
SSH	*ES*	*AVG*	48.4±1.67	30.9±1.18	45.5±2.03	24.7±0.45
SSH	*MPS*	*AVG*	**50.8±2.07**	**35.4±1.31**	**47.5±2.34**	**27.4±0.37**
SSH	$P_\triangle$	*AVG*	43.5±1.30	24.6±1.13	39.9±2.18	13.9±0.27
SSH	*ES*	*NC*	59.26± 1.76	64.17±1.3	53.52±1.96	57.4±0.27
SSH	*MPS*	*NC*	**63.4±1.67**	**69.37±1.45**	**59.59±1.59**	**61.1±0.33**
SSH	$P_\triangle$	*NC*	60.99±1.52	65.75±1.27	57.54±1.61	57.3±0.50
SSH	*ES*	*NN*	68.6±1.53	75.3±1.56	62.7±1.82	69.3±0.40
SSH	*MPS*	*NN*	**71.7±1.76**	**78.9±1.61**	67.7±2.75	**72.1**±0.30
SSH	$P_\triangle$	*NN*	71.6±1.19	78.7±1.23	**67.9±2.75**	**72.1±0.44**
SSNRA	*ES*	*AVG*	65.3±1.15	31.7±1.31	39.1±1.21	25.5±0.24
SSNRA	*MPS*	*AVG*	**65.9±1.24**	**31.8±0.69**	**41.7±2.50**	**27.6±0.33**
SSNRA	$P_\triangle$	*AVG*	56.5±1.43	20.1±0.63	38.4±1.87	12.4±0.3
SSNRA	*ES*	*NC*	74.9±1.09	62.89±1.07	62.31±2.76	56.2±0.36
SSNRA	*MPS*	*NC*	**76.34±1.09**	**65.15±1.79**	**62.54±2.11**	**59.1±0.45**
SSNRA	$P_\triangle$	*NC*	74.44±1.76	61.49±1.55	60.35±1.99	54.7±0.61
SSNRA	*ES*	*NN*	**82.8±1.34**	75.8±1.47	**72.8±1.99**	68.6±0.31
SSNRA	*MPS*	*NN*	82.6±0.77	77.1±1.12	72.0±2.09	**70.8±0.46**
SSNRA	$P_\triangle$	*NN*	**82.8±1.44**	**77.2±1.04**	71.6±1.50	70.8±0.40
SSNRSD	*ES*	*AVG*	42.5±1.03	27.5±0.60	38.1±1.34	22.8±0.36
SSNRSD	*MPS*	*AVG*	**45.6±1.08**	**29.4±1.57**	**40.9±1.47**	**24.2±0.33**
SSNRSD	$P_\triangle$	*AVG*	36.8±1.30	19.0±1.05	34.7±1.88	12.4±0.26
SSNRSD	*ES*	*NC*	60.25±1.21	51.17±1.37	48.98±2.18	44.0±0.45
SSNRSD	*MPS*	*NC*	**64.1±1.78**	**53.8±1.36**	**57.78±2.06**	**46.8±0.38**
SSNRSD	$P_\triangle$	*NC*	61.8±1.16	49.38±1.43	51.81±2.32	42.1±0.40
SSNRSD	*ES*	*NN*	68.1±1.55	61.6±1.23	56.3±2.81	54.8±0.42
SSNRSD	*MPS*	*NN*	**71.2±0.98**	**64.7±1.16**	**59.3±3.00**	57.1±0.52
SSNRSD	$P_\triangle$	*NN*	70.7±2.32	65.0±1.44	58.6±2.25	**57.4±0.47**

Table C.2: Summary of 10-fold cross-validation evaluation on LANGG68, COIL-100, ETH80 and ALOI-1000 datasets trained using incremental learning

Feature space	Similarity measure	Classifier type	Avg. acc. ± stdev (%)			
			LANGG68	COIL-100	ETH80	ALOI-1000
AREA	*ES/MPS*	*AVG*	28.9±1.82	10.5±1.13	39.6±2.18	**22.0±0.28**
AREA	$P_\triangle$	*AVG*	**31.6±1.44**	**11.1±0.69**	**40.0±2.65**	4.8±0.23
AREA	*ES/MPS*	*NC*	25.83±1.25	8.09±0.86	36.41±2.07	**20.2±0.19**
AREA	$P_\triangle$	*NC*	28.42±**1.96**	**10.79±0.67**	**38.77±3.12**	12.4±0.51
AREA	*ES/MPS*	*NN*	25.5±1.31	**7.3±0.67**	**36.2±2.03**	15.2±0.22
AREA	$P_\triangle$	*NN*	**25.2±1.40**	7.0±0.64	35.5±2.52	**15.2±0.44**
RADSD	*ES/MPS*	*AVG*	19.2±1.86	6.1±0.74	31.4±1.92	**2.1±0.12**
RADSD	$P_\triangle$	*AVG*	**23.4±1.05**	**7.6±0.87**	**36.1±3.26**	1.7±0.07
RADSD	*ES/MPS*	*NC*	16.62±1.14	5.31±0.61	30.38±2.57	**1.6±0.1**
RADSD	$P_\triangle$	*NC*	**18.61±1.6**	**6.85±0.85**	**30.34±2.85**	1.3±0.08
RADSD	*ES/MPS*	*NN*	16.3±1.23	**4.9±0.56**	**28.6±1.82**	**1.9±0.11**
RADSD	$P_\triangle$	*NN*	**16.4±1.41**	4.7±0.84	27.7±2.75	**1.9±0.11**
SLH	*ES*	*AVG*	53.7±1.17	33.9±1.35	45.1±3.51	21.0±0.33
SLH	*MPS*	*AVG*	**58.9±1.31**	**39.4±0.56**	**54.1±1.61**	**23.0±0.35**
SLH	$P_\triangle$	*AVG*	49.8±1.87	26.7±1.02	44.3±3.39	12.8±0.35
SLH	*ES*	*NC*	59.26±1.17	52.12±1.37	53.82±3.01	40.7±0.37
SLH	*MPS*	*NC*	**62.46±1.58**	**56.89±1.63**	**56.32±1.63**	**42.5±0.51**
SLH	$P_\triangle$	*NC*	59.75±1.75	54.29±1.42	54.88±2.47	39.7±0.30
SLH	*ES*	*NN*	65.2±1.28	60.4±1.13	59.3±4.28	49.1±0.51
SLH	*MPS*	*NN*	67.8±1.56	**65.2±2.02**	**59.4±2.21**	**50.3±0.42**
SLH	$P_\triangle$	*NN*	**67.9±1.56**	64.0±2.03	58.9±2.90	50.2±0.57
SSH	*ES*	*AVG*	50.3±1.72	34.2±1.09	47.5±2.48	24.0±0.32
SSH	*MPS*	*AVG*	**53.7±1.04**	**39.9±1.79**	**52.5±1.85**	**27.2±0.52**
SSH	$P_\triangle$	*AVG*	43.5±1.70	25.2±0.57	50.2±2.86	13.6±0.4
SSH	*ES*	*NC*	54.95±1.72	60.1±1.44	51.9±2.78	53.7±0.18
SSH	*MPS*	*NC*	**58.75±1.04**	**65.16±1.99**	**58.42±2.06**	**57.4±0.29**
SSH	$P_\triangle$	*NC*	55.68±1.70	61.26±1.31	55.98±1.97	53.4±0.46
SSH	*ES*	*NN*	62.1±2.24	69.7±1.39	58.3±1.99	64.2±0.29
SSH	*MPS*	*NN*	64.9±1.72	**74.3±1.33**	**63.1±3.38**	**67.0±0.26**
SSH	$P_\triangle$	*NN*	**65.4±1.43**	73.7±1.48	62.1±1.94	**67.0±0.42**
SSNRA	*ES*	*AVG*	64.7±0.92	33.9±1.74	61.0±1.62	25.5±0.50
SSNRA	*MPS*	*AVG*	**65.5±1.37**	**35.3±1.38**	**62.2±1.88**	**28.4±0.70**
SSNRA	$P_\triangle$	*AVG*	53.6±1.71	19.9±1.01	52.13±2.27	11.2±0.33
SSNRA	*ES*	*NC*	72.76±1.62	59.65±2.15	**62.09±3.26**	52.4±0.31
SSNRA	*MPS*	*NC*	**73.17±1.54**	**62.41±2.04**	61.63±1.4	**55.5±0.43**
SSNRA	$P_\triangle$	*NC*	70.42±1.95	58.49±1.6	59.5±1.76	51.0±0.48
SSNRA	*ES*	*NN*	**78.4±1.28**	71.9±1.33	68.0±2.35	64.0±0.45
SSNRA	*MPS*	*NN*	78.3±0.76	71.3±1.36	**68.8±2.38**	65.8±0.46
SSNRA	$P_\triangle$	*NN*	77.8±0.72	**72.1±1.51**	67.9±2.89	**65.9±0.45**
SSNRSD	*ES*	*AVG*	44.4±1.69	28.5±1.35	46.6±3.41	23.4±0.29
SSNRSD	*MPS*	*AVG*	**48.2±1.99**	**32.3±0.99**	**49.4±2.45**	**25.1±0.24**
SSNRSD	$P_\triangle$	*AVG*	38.5±2.00	19.8±0.64	36.8±3.06	13.1±0.33
SSNRSD	*ES*	*NC*	55.96±1.22	48.2±1.33	46.09±2.57	41.1±0.38
SSNRSD	*MPS*	*NC*	**56.61±2.43**	**50.42±1.18**	**50.57±3.11**	**43.3±0.38**
SSNRSD	$P_\triangle$	*NC*	56.11±1.8	45.2±1.41	47.96±2.79	38.7±0.37
SSNRSD	*ES*	*NN*	62.1±1.91	57.3±1.89	51.5±2.77	50.6±0.27
SSNRSD	*MPS*	*NN*	**65.5±1.96**	60.3±2.30	**55.3±2.72**	52.6±0.27
SSNRSD	$P_\triangle$	*NN*	64.6±0.98	**60.4±1.49**	54.2±2.95	**52.8±0.46**

Table C.3: Summary of results obtained with different MCML configurations on COIL-100 and ETH80 datasets

Dataset	Classifiers	Classifier used for prediction	Classifier ranking	Forgetting	Stopping condition	#exps.	Avg. #cats learned	Avg. #iters	Avg. #inst/cat	Avg. protocol acc.(%)	Global acc.(%)
COIL-100											
	Only *SSNRA-MPS-NN*	*SSNRA-MPS-NN*	Not required	No	Lack of data	5	39.4	1026.4	10.4	58.5	64.1
	All base classifiers	*DS5TOP*	Dynamic	No	All cats. learned	1	100.0	2263.0	7.9	66.0	69.4
					Lack of data	4	80.8	1817.3	8.0	63.2	68.4
	All classifiers except *MAJORITY-ALL*	Current best	Dynamic	No	Lack of data	5	77.2	1683.8	7.9	64.0	68.4
	All classifiers	*MAJORITY-ALL*	Dynamic	No	Lack of data	5	79.4	1797.4	8.2	63.8	68.0
	All classifiers	Current best	Dynamic	Yes	Lack of data	5	73.4	1745.2	6.1	62.2	67.3
	All classifiers	Current best	Static	No	All cats. learned	1	100.0	2321.0	8.0	64.3	69.1
					Lack of data	4	73.5	1681.5	8.1	62.6	67.8
	All classifiers	Current best	Dynamic	No	All cats. learned	1	100.0	2319.0	8.3	63.2	68.4
					Lack of data	4	64.3	1385.0	7.8	63.6	68.1
ETH80											
	Only *SSNRA-MPS-NN*	*SSNRA-MPS-NN*	Not required	No	All cats. learned	3	8.0	162.3	10.0	50.4	56.4
					Breakpoint	2	7.0	43.5	3.07	65.1	68.3
	All base classifiers	*DS5TOP*	Dynamic	No	All cats. learned	5	8.0	121.6	7.2	53.5	59.6
	All classifiers except *MAJORITY-ALL*	Current best	Dynamic	No	All cats. learned	5	8.0	132.6	8.0	54.2	59.0
	All classifiers	*MAJORITY-ALL*	Dynamic	No	All cats. learned	5	8.0	121.6	7.1	56.0	61.5
	All classifiers	Current best	Dynamic	Yes	All cats. learned	5	8.0	150.8	5.8	51.0	57.3
	All classifiers	Current best	Static	No	All cats. learned	5	8.0	122.2	7.4	52.4	59.4
	All classifiers	Current best	Dynamic	No	All cats. learned	5	8.0	141.6	8.3	53.7	59.9

Bibliography

D. W. Aha. *Lazy learning.* Kluwer Academic Publishers, 1997.

D. W. Aha and D. Wettschereck. Case-based learning: Beyond classification of feature vectors. In M. van Someren and G. Widmer, editors, *Proceedings of the Ninth European Conference on Machine Learning (ECML 1997), LNCS*, volume 1224, pages 329–336. London, UK: Springer-Verlag, 1997.

D. W. Aha, D. Kibler, and M. Albert. Instance based learning algorithms. *Machine Learning*, 6:37–66, 1991.

A. Al-Ani and M. Deriche. A new technique for combining multiple classifiers using the dempster-shafer theory of evidence. *Lournal of Artificial Intelligence Research*, 17:333–361, 2002.

M. L. Anderson and D. R. Perlis. *Symbol systems.* London, UK: Macmillan, 2002.

A. Andreopoulos and J. K. Tsotsos. 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding*, 117(8):827 – 891, 2013.

M. Antunes and L. S. Lopes. Unsupervised internet-based category learning for object recognition. In *Image Analysis and Recognition*, pages 766–773. Springer, 2013.

F. G. Ashby and L. A. Alfonso-Reese. Categorization as probability density estimation. *Journal of Mathematical Psychology*, 39:216–233, 1995.

F. G. Ashby and J. B. O'Brien. Category learning and multiple memory systems. *Trends in Cognitive Science*, 9:83–89, 2005.

M.-F. Balcan and A. Blum. On a theory of learning with similarity functions. In W. W. Cohen and A. Moore, editors, *ICML*, volume 148 of *ACM International Conference Proceeding Series*, pages 73–80. ACM, 2006.

D. H. Ballard and C. Yu. A multimodal learning interface for word acquisition. In *International conference on Acoustics, Speech and Signal Processing*, 2003.

L. Barsalou. Perceptual symbol systems. *Behavioral and Brain Sciences*, 22(4):577–609, 1999.

L. W. Barsalou, W. K. Simmons, A. K. Barbey, and C. D. Wilson. Grounding conceptual knowledge in the modality-specific systems. *Trends in Cognitive Science*, 7:84–91, 2003.

J. Batali. *Computational simulations of the emergence of grammar*, pages 405–426. Cambridge University Press, Cambridge, 1998.

E. Bates, D. Thal, B. Finlay, and B. Clancy. *Early Language Development And Its Neural Correlates*, volume 7, pages 69–110. Elsevier, Amsterdam, 1992.

P. Baxter, T. Belpaeme, L. Canamero, P. Cosi, Y. Demiris, and V. Enescu. Long-term human-robot interaction with young users. In *IEEE/ACM Human-Robot Interaction 2011 Conference (Robots with Children Workshop)*, 2011.

H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. ISSN 1077-3142.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 2002.

T. Belpaeme and S. J. Cowley. Extending symbol grounding. *Interaction Studies*, 8:1–6, 2007.

I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

P. Bloom. *How Children Learn the Meanings of Words*. MIT Press, Cambridge, 2000.

P. Bloom. Word learning. *Current Biology*, 11:5–6, 2001.

D. Bohus and E. Horwitz. Dialog in the open world: Platform and applications. In *ICMI'2009*, MA, 2009.

P. C. Bomba and E. R. Siqueland. The nature and structure of infant form categories. *Journal of Experimental Child Psychology*, 37:609–636, 1983.

A. Bosch, A. Zisserman, and X. Muoz. Representing shape with a spatial pyramid kernel. In N. Sebe and M. Worring, editors, *CIVR*, pages 401–408. ACM, 2007. ISBN 978-1-59593-733-9.

M. A. Branch and A. Grace. *MATLAB: optimization toolbox: user's guide version 1.5*. The MathWorks, 1996.

C. Breazeal. Toward sociable robots. *Robotics and Autonomous Systems*, 42(3):167–175, 2003.

T. Briscoe. The acquisition of grammar in an evolving population of language agents. *Machine Intelligence: Electronic Transactions in Artificial Intelligence*, 16, 1999.

R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

R. A. Brooks. Intelligence without representation. *Artificial Intelligence Journal*, 47:139–159, 1991.

A. Cangelosi. *Approaches to grounding symbols in perceptual and sensorimotor categories*, pages 719–737. Oxford: Elsevier Science, 2005.

A. Cangelosi and S. Harnad. The adaptive advantage of symbolic theft over sensorimotor toil: Grounding language in perceptual categories. *Evolution of Communication*, 4(1):17–142, 2000.

S. Carey. *The child as word learner*, pages 264–293. MIT Press, 1978.

A. Chauhan and L. S. Lopes. Manhattan-pyramid distance: A solution to an anomaly in pyramid matching by minimization. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2668–2672. IEEE, 2012.

A. Chauhan and L. Seabra Lopes. Acquiring vocabulary through human robot interaction: a learning architecture for grounding words with multiple meanings. In *AAAI Fall Symposium (DwR 2010)*, Arlington, VA, USA, 2010a.

A. Chauhan and L. Seabra Lopes. Aiding categorization by grounding spoken words - an infant inspired approach to concept formation and language acquisition. In *4th international conference on Brain-Inspired Cognitive Systems (BICS 2010)*, Madrid, Spain, 2010b.

A. Chauhan, A. Nascimento, B. Werneck, and L. Seabra Lopes. Embodied language acquisition: a proof of concept. In *Progress in Artificial Intelligence: 14th Portuguese Conference on Artificial Intelligence - EPIA'2009*. Springer, 2009.

A. Chella, H. Dindo, and D. Zambuto. Grounded human-robot interaction. In *AAAI-2009 Fall Symposium on Biologically Inspired Cognitive Architectures*, pages 33–38, 2009.

N. Chomsky. *Syntactic Structures.* The Hague: Mouton, 1957.

N. Chomsky. A review of B. F. Skinner's verbal behavior. *Language*, 35(1):26–58, 1959.

N. Chomsky. *Aspects of the theory of syntax.* MA: MIT Press, 1965.

N. Chomsky. *Reflections on Language.* Pantheon: New York., 1975.

A. Clark. *Being there: Putting brain, body and world together again.* Cambridge, MA: MIT Press, 1997.

J. Connell, E. Marcheret, S. Pankanti, M. Kudoh, and R. Nishiyama. An extensible language interface for robot manipulation. In *Proc. Artificial General Intelligence Conf. (AGI-12)*, volume LNAI 7716, pages 21–30, 2012.

S. J. Cowley. *The Emergence and Evolution of Linguistic Communication*, chapter Distributed language: Biomechanics, functions and the origins of talk, pages 105–129. London: Springer, 2006.

S. J. Cowley. How human infants deal with symbol grounding. *Interaction Studies*, 8:83–104, 2007.

M. T. Cox. Metacognition in computation: A selected research review. *Artificial Intelligence*, 169:104–141, 2005.

D. Crystal. How many words? *English Today*, 12:11–14, 1987.

C. Cunningham, R. Weber, J. M. Proctor, C. Fowler, and M. Murphy. Investigating graphs in textual case-based reasoning. In *ECBR2004*, pages 573–586, 2004.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. L. 0002. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009. ISBN 978-1-4244-3992-8.

R. Diestel. *Graph Theory*. Springer, Heidelberg, 2000.

H. Dindo and D. Zambuto. Resolving ambiguities in a grounded human-robot interaction. In *18th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 408–419, 2009.

H. Dindo and D. Zambuto. A probabilistic approach to learning a visually grounded language model through human-robot interaction. In *2010 IEEE/RSJ International Conference on Intelligent RObots and Systems (IROS)*, pages 790–796, October 2010.

W. Dong, Z. Wang, M. Charikar, and K. Li. Efficiently matching sets of features with random histograms. In A. El-Saddik, S. Vuong, C. Griwodz, A. D. Bimbo, K. S. Candan, and A. Jaimes, editors, *ACM Multimedia*, pages 179–188. ACM, 2008. ISBN 978-1-60558-303-7.

E. Dresher and J. Kaye. A computational learning model for metrical phonology. *Cognition*, 34:137–195, 1990.

R. P. W. Duin and D. M. J. Tax. Experiments with classifier combining rules. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2000. ISBN 3-540-67704-6.

J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

J. L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–225, 1991.

J. L. Elman. An alternative view of mental lexicon. *Trends in Cognitive Science*, 8:301–306, 2004.

J. L. Elman, E. A. Bates, M. H. Johnson, A. Karmiloff-Smith, and K. Parisi, D. Plunkett. *Rethinking Innateness A connectionist perspective on development.* MA: MIT Press, 1996.

N. Evans and S. C. Levinson. The myth of langauge universals: language diversity and its importance for cognitive science. *Behavioral and Brain Sciences*, 32:429–448, 2009. doi: 10.1017/S0140525X0999094X.

M. Everingham, L. Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal on Computer Vision*, 88:303–338, 2010.

L. Fenson, P. S. Dale, J. S. Reznick, E. Bates, D. Thal, and S. Pethick. Variability in early communicative development. *Monographs of the Society for Research in Child Development*, 59(5), 1994.

W. T. Fitch. Unity and diversity in human language. *Philosophical Transactions of The Royal Society B*, 366(1563):376–388, 2011. doi: 10.1098/rstb.2010.0223.

W. T. Fitch, M. D. Hauser, and N. Chomsky. The evolution of the language faculty: Clarifications and implications. *Cognition*, 97(2):179–210, September 2005. doi: 10.1016/j.cognition.2005.02.005.

J. H. Flavell. First discussants comments: What is memory development the development of? *Human Development*, 14:272–278, 1971.

J. Fodor. Unambiguous triggers. *Liguistic Inquiry*, 29(1):1–36, 1998.

T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots: Concepts, design, and applications. *Robotics and Autonomous Systems*, 42:143–166, 2003.

A. Francis and A. Ram. The utility problem in case-based reasoning. In *Workshop on Case Based Reasoning*, pages 160–167. Washington, DC: AAAI Press, 1993.

M. Fritz, G. Kruijff, and B. Schiele. Tutor-based learning of visual categories using different levels of supervision. *Computer Vision and Image Understanding*, 114:564–573, 2010.

J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulder. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

E. Gibson and K. Wexler. Triggers. *Liguistic Inquiry*, 25(3):407–454, 1994.

J. J. Gibson. *The ecological approach to visual perception.* Boston, MA: Houghton Mifflin, 1979.

P. E. Gill, W. Murray, and M. H. Wright. Practical optimization. 1981.

P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Procedures for optimization problems with a mixture of bounds and general linear constraints. *ACM Transactions on Mathematical Software (TOMS)*, 10(3):282–298, 1984.

J. Gillette, H. Gleitman, L. Gleitman, and A. Lederer. Human simulations of vocabulary learning. *Cognition*, 73:135–176, 1999.

L. R. Gleitman and B. Landau. *The acquisition of the Lexicon.* MA: MIT Press, 1994.

E. Gold. Language identification in limit. *Information and Control*, 10:447–474, 1967.

K. Gold, M. Doniec, C. Crick, and B. Scassellati. Robotic vocabulary building using extension inference and implicit contrast. *Artificial Intelligence*, 173(1):145–166, 2009.

M. Goodrich and A. Schultz. Human-robot interaction: a survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.

K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, pages 1458–1465. IEEE Computer Society, 2005.

K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, 8:725–760, 2007.

A. Greco, T. Riga, and A. Cangelosi. The acquisition of new categories through grounded symbols: An extended connectionist model. In O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, editors, *Artificial Neural Networks and Neural Information Processing ICANN/ICONIP 2003*, pages 773–770. Springer, 2003.

J. Greenberg. *Universals of langauge.* Cambridge, MA: MIT Press, 1963.

S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

S. Harnad, S. J. Hanson, and J. Lubin. Categorical perception and the evolution of supervised learning in neural nets. In D. W. Powers and L. Reeker, editors, *AAAI Spring Symposium on Machine Learning of Natural Language and Ontology*, pages 65–74, 1991.

S. Harnad, S. Hanson, and J. Lubin. Learned categorical perception in neural nets: Implications for symbol grounding. In V. Honavar and L. Uhr, editors, *Symbol processors and connectionist models in artificial intelligence and cognitive modelling: Steps toward principled integration*, pages 191–206. San Diego, CA: Academic Press, 1995.

M. D. Hauser. Et tu homo sapiens? review of the cultural origins of human cognition". *Science*, 288(5467):816–817, 2000.

M. D. Hauser, N. Chomsky, and T. W. Fitch. The faculty of lanaguge: What is it, who has it, and how did it evolve? *Science*, 298(5598):1569–1579, 2002.

J. Hawkins. *Explaining langauge universals.* Oxford, UK: Basil Blackwell, 1990.

F. Hegel, M. Lohse, A. Swadzba, S. Wachsmuth, K. Rohlfing, and B. Wrede. Classes of applications for social robots: a user study. In *International Symposium on Robot and Human Interactive Communication*, pages 938–943, 2007.

C. F. Hockett. *Logical considerations in the study of animal communication*, pages 392–430. Washington, DC: American Institute of Biological Sciences, 1960.

C. F. Hockett. The problem of universals in language. In J. Greenberg, editor, *Universals of language*, pages 1–29. Cambridge, MA: MIT Press, 1966.

R. S. Holt, P. A. Mastromarino, E. K. Kao, and M. B. Hurley. Information theoretic approach for performance evaluation of multi-class assignment systems. In I. Kadar, editor, *Signal Proc., Sensor Fusion, and Target Recognition XIX (SPIE)*, volume 7697:76970R/1-12, 2010.

N. Iwahashi. Robots that learn language: Developmental approach to human-machine conversations. In P. Vogt and et al., editors, *Symbol Grounding and Beyond: Proceedings of the Third International Workshop on the Emergence and Evolution of Linguistic Communication*, LNCS, pages 143–167. Springer, 2006.

N. Iwahashi, K. Sugiura, R. Taguchi, T. Nagai, and T. Taniguchi. Robots that learn to communicate: A developmental approach to personally and physically situated human-robot conversations. In *AAAI Fall Symposium, Dialog with Robots(FS-10-05)*, pages 38–43, 2010.

R. Jackendoff. *Foundations of language: Brain, Meaning, Grammar, Evolution.* Oxford University Press, 2002.

N. Japkowicz. Are we better off without counter-examples? In *First International ICSC Congress on Computational Intelligence Methods and Applications (CIMA-99)*, pages 242–248, 1999.

S. Kapur and R. Clark. The automatic construction of a symbolic parser via statistical techniques. In *ACM Linguistics Workshop on Integration of Statistical and Symbolic Systems*, 1994.

W. Kennedy and J. G. Trafton. Long-term symbolic learning. *Cognitive Systems Research*, 8:237–247, 2007.

W. G. Kennedy and K. D. Jong. Characteristics of long-term learning in soar and its application to the utility problem. In *Twentieth International Conference on Machine Learning*, pages 337–344, 2003.

S. Kirby and J. Hurford. Simulating the evolution of language. In A. Cangelosi and D. Parisi, editors, *The Emergence of Linguistic Structure: An overview of the Iterated learning model*, pages 21–148. Springer, Heidelberg, 2002.

S. Kirstein and H. Wersing. A biologically inspired approach for interactive learning of categories. In *Proceedings of the Tenth International Conference on Development and Learning (ICDL 2011)*, pages 1–6, 2011.

S. Kirstein, H. Wersing, H.-M. Gross, and E. Krner. A life-long learning vector quantization approach for interactive learning of multiple categories. *Neural Networks*, 28:90–105, 2012.

J. Kittler, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:226–239, 1998.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artifcial Intelligence (IJCAI)*, page 11371145, 1995.

V. Krunic, G. Salvi, A. Bernardino, L. Montesano, and J. Santos-Victor. Affordance based word-to-meaning association. In *IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009.

J. K. Kruschke. *Category learning*, chapter 7, pages 183–201. London, UK: Sage Publications, 2005.

B. Landau, L. B. Smith, and S. Jones. The importance of shape in early lexical learning. *Cognitive Development*, 3:299–321, 1988.

P. Langley, J. E. Laird, and S. Rogers. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160, 2009.

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR (2)*, pages 2169–2178. IEEE Computer Society, 2006. ISBN 0-7695-2597-0.

Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng. Building high-level features using large scale unsupervised learning. In *International Conference in Machine Learning*, 2012.

B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *International Conference on Computer Vision and Pattern Recognition 2003 (CVPR03)*, 2003.

V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

S. E. Levinson, K. Squire, R.-S. Lin, and M. McClain. Automatic language acquisition by an autonomous robot. In *AAAI Spring Symposium on Developmental Robotics*, 2005.

S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

N. Love. Cognition and the language myth. *Language Sciences*, 26(6):525 – 544, 2004.

A. Lovett, M. Dehghani, and K. Forbus. Incremental learning of perceptual categories for open-domain sketch recognition. In M. M. Veloso, editor, *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 447–452, 2007.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

K. F. MacDorman, K. Tatani, Y. Miyazaki, and M. Koeda. Proto-symbol emergence. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, pages 1619–1625, Takamatsu, Japan, 2000.

R. Mairal and J. Gil. *Linguistic universals.* Cambridge, UK: Cambridge University Press, 2006.

C. D. Manning, P. Raghavan, and H. Schtze. *An introduction to information retrieval.* Cambridge University Press, Cambridge, 2008. ISBN 9780521865715 0521865719.

G. Marcus. *Poverty of the stimulus arguments.* Cambridge, MA: MIT Press, 1999.

E. S. Markman. *Categorization and naming in children.* MIT Press, Cambridge, MA, 1989.

S. Markovitch and P. D. Scott. The role of forgetting in learning. In *Fifth International Conference on Machine Learning*, pages 459–465, 1988.

J. Mehler, N. Sebástian-Gallés, and M. Nespor. *Biological foundations of language: language acquisition, cues for parameter setting and the bilingual infant*, pages 825–836. Cambridge, MA: MIT Press, 2004.

G. Mensink and J. G. Raaijmakers. A model for interference and forgetting. *Psychological Review*, 95:434–455, 1988.

D. J. Messer. *The Development of Communication.* Chichester, England: Wiley, 1994.

L. Meteyard and G. Vigliocco. *The role of sensory and motor information in semantic perception*, pages 293–312. London: Elsevier, 2008.

L. Meteyard, S. Rodriguez Cuardrado, B. Bahrami, and G. Vigliocco. Coming of age: a review of embodiment and the neuroscience of semantics. *Cortex (Special Issue: Language and the Motor System)*, 48(7):788–804, 2012.

S. Minton. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42:363–391, 1990.

M. Montaner, B. Lpez, and J. L. de la Rosa. Improving case representation and case base maintenance in recommender agents. In *ECCBR 2002 (LNCS)*, volume 2416, pages 234–248. Berlin: Springer, 2002.

L. Montesano, M. Lopes, A. Bernadino, and J. Santos Victor. Learning object affordances: From sensory-motor maps to imitation. *IEEE Transactions on Robotics, Special Issue on Bio-Robotics*, 24(1):15–26, 2008.

R. Mooney. The effect of rule use on the utility of explanation-based learning. In *Eleventh International Joint Conference on Artificial Intelligence (IJCAI 1989)*, pages 725–730, 1989.

R. Murphy, T. Nomura, A. Billard, and J. Burke. Humanrobot interaction. *IEEE Robotics & Automation Magazine*, 17:85–89, 2010. doi: 10.1109/MRA.2010.936953.

B. Mutlu, T. Shiwa, T. Kanda, H. Ishiguro, and N. Hagita. Footing in human-robot conversations: How robots might shape participant roles using gaze cues. In *HRI 2009*, 2009.

G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1): 31–88, 2001.

T. O. Nelson and L. Narens. Metamemory: A theoretical framework and new findings. In G. H. Bower, editor, *The Psychology of Learning and Motivation: Advances in Research and Theory*, volume 26, pages 125–169. Academic Press, 1990.

S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-100). Cucs-006-96, Columbia University, 1996.

M. Nespor, M. Peña, and J. Mehler. On the different roles of vowels and consonants in speech processing and language acquisition. *Lingue e Linguaggio*, 2:221–247, 2003.

A. Newell. Physical symbol systems. *Cognitive Science*, 4:135–183, 1980.

A. Newell and H. A. Simon. *Human problem solving.* Englewood cliffs, New Jersey: Prentice-Hall Inc., 1972.

N. J. Nilsson. Human-level artificial intelligence? be serious! *AI Magazine*, 26(4):68–75, 2005.

M. A. Nowak, J. Plotkin, and D. Krakauer. The evolutionary language game. *Journal of Theoretical Biology*, 200:147–162, 1999.

A. Onwuegbuzie and N. Leech. Validity and qualitative research: An oxymoron? *Quality & Quantity: International Journal of Methodology*, 41(2):233–249, 2007.

M. Pagel. Human language as a culturally transmitted replicator. *Nature Reviews Genetics*, 10:405–415, 2009.

T. Pavlidis. A review of algorithms for shape analysis. *Computer Graphics and Image Processing*, 7(2):243 – 258, 1978. ISSN 0146-664X.

M. Peña, L. L. Bonatti, M. Nespor, and J. Mehler. Signal driven computations in speech processing. *Science*, 298(5593):604–607, 2002.

R. Pereira and L. Seabra Lopes. *Learning Visual Object Categories with Global Descriptors and Local Features*, volume 5816 of *Lecture Notes in Computer Science*, pages 225–236. Springer Berlin / Heidelberg, 2009.

J. W. Pilley and A. K. Reid. Border collie comprehends object names as verbal referents. *Behavioral Processes*, 2010.

S. Pinker. *Language Learnability and Language Development.* MA: Harvard University Press, 1984.

S. Pinker. *The Language Instinct.* William Morrow and Company, NY, 1994.

S. Pinker and R. Jackendoff. The faculty of language: What's special about it? *Cognition*, 95(2):201–236, 2005.

J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft research, 1998.

K. Plunkett and C. Sinha. Connectionism and developmental theory. *British Journal of Developmental Psychology*, 10:209–254, 1992.

K. Plunkett, C. Sinha, M. F. Moller, and O. Strandsby. Symbol grounding or the emergence of symbols? vocabulary growth in children and a connectionist net. *Connection Science*, 4(3-4):292–312, 1992.

R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, 31:497–508, 2001.

Z. W. Pylyshyn. *Computation and Cognition: Toward a Foundation for Cognitive Science.* London, UK: MIT Press, 1985.

X. Qi and B. D. Davison. Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41:12:1–12:31, February 2009. ISSN 0360-0300. doi: 10.1145/1459352.1459357.

M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Computer Vision and Pattern Recognition (CVPR)*, 2003.

P. Refaeilzadeh, L. Tang, and H. Liu. *Cross Validation.* Springer, 2009.

T. Regier, B. Corrigan, C. Cabasaan, A. Woodward, M. Gasser, and L. Smith. The emergence of words. In *Proceedings of the Twenty-third Annual Conference of the Cognitive Science Society*, pages 815–820, 2001.

E. Rosch. On the internal structure of perceptual and semantic categories. In T. E. Moore, editor, *Cognitive Development and the Acquisition of Language*, pages 111–144. Academic Press, New York, NY, 1973.

D. Roy. Grounded spoken language acquisition: Experiments in word learning. *IEEE Transactions on Multimedia*, 5(2):197–209, 2003.

D. Roy. Grounding words in perception and action: computational insights. *Trends in Cognitive Science*, 9:389396, 2005.

D. Roy and A. Pentland. Learning words from sights and sounds: A computational model. *Cognitive Science*, 26:113–146, 2002.

D. E. Rumelhart. *Some problems with the notion that words have literal meanings*, pages 71–82. Cambridge University Press, 1979.

B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3): 157–173, 2008.

H. Sahbi, J.-Y. Audibert, and R. Keriven. Context-dependent kernels for object classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(4):699–708, 2011.

N. J. Sales and R. G. Evans. An approach to solving the symbol grounding problem: Neural networks for object naming and retrieval. In *International Conference on Cooperative Multimodal Communications (CMC-95).*, Eindhoven, The Netherlands, 1995.

J. C. Schlimmer and D. Fisher. A case study of incremental concept induction. In *AAAI*, pages 496–501, 1986.

J. C. Schlimmer and R. H. Granger. Beyond incremental processing: Tracking concept drift. In T. Kehler, editor, *AAAI*, pages 502–507. Morgan Kaufmann, 1986.

L. Seabra Lopes. Carl: From situated activity to language level interaction and learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 890–896, 2002.

L. Seabra Lopes and A. Chauhan. *Workshop on External Symbol Grounding, Book of Abstracts and Papers*, chapter One-Class Lifelong Learning Approach to Grounding, pages 15–23. 2006.

L. Seabra Lopes and A. Chauhan. How many words can my robot learn? an approach and experiments with one-class learning. *Interaction Studies*, 8(1):53–81, 2007.

L. Seabra Lopes and A. Chauhan. Open-ended category learning for language acquisition. *Connection Science*, 20(4):277–297, 2008. doi: 10.1080/09540090802413228.

L. Seabra Lopes and J. H. Connell. Semisentient robots: Routes to integrated intelligence. *IEEE Intelligent Systems*, 16(5):10–14, 2001.

L. Seabra Lopes and A. Teixeira. Human-robot interaction through spoken language dialogue. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 528–534, Kagawa University, Takamatsu, Japan, 2000.

L. Seabra Lopes and Q. H. Wang. Towards grounded human-robot communication. In *11th IEEE International Workshop on Robot and Human Interactive Communication*, pages 312–318, 2002.

L. Seabra Lopes, A. Chauhan, and J. Silva. Towards long-term visual learning of object categories in human-robot interaction. In J. M. Neves, M. Santos, and J. Machado, editors, *New Trends in Artificial Intelligence*, pages 623–634. Associao Portuguesa para a Inteligncia Artificial, 2007.

J. R. Searle. Minds, brains and programs. *Behavioral and Brain Sciences*, 3:417–457, 1980.

G. Shafer. *A Mathematical Theory of Evidence.* Princeton, NJ: Princeton University Press, 1976.

H. A. Simon. Information processing models of cognition. *Annual Review of Psychology*, 30: 363–396, 1979.

D. Skočaj, G. Berginc, B. Ridge, A. Štimec, M. Jogan, O. Vanek, A. Leonardis, M. Hutter, and N. Hawes. A system for continuous learning of visual concepts. In *International Conference on Computer Vision Systems ICVS 2007*, Bielefeld, Germany, 2007.

D. Skočaj, M. Kristan, and A. Leonardis. Continuous learning of simple visual concepts using incremental kernel density estimation. In A. Ranchordas and H. Arajo, editors, *VISAPP (1)*, pages 598–604. INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2008. ISBN 978-989-8111-21-0.

L. B. Smith and L. Samuelson. An attentional learning account of the shape bias: reply to cimpian and markman (2005) and booth, waxman, and huang (2005). *Developmental Psychology*, 42(6):1339–1343, 2006.

M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45:427–437, 2009.

T. P. Spexard, M. Hanheide, and G. Sagerer. Human-oriented interaction with an anthropomorphic robot. *IEEE Transactions on Robotics*, 23:852–862, 2007.

N. Srebro. How good is a kernel when used as a similarity measure? In N. H. Bshouty and C. Gentile, editors, *COLT*, volume 4539 of *Lecture Notes in Computer Science*, pages 323–335. Springer, 2007.

L. Steels. Language games for autonomous robots. *IEEE Intelligent Systems*, 16(5):16–22, 2001.

L. Steels. Evolving grounded communication for robots. *Trends in Cognitive Science*, 7(7): 308–312, 2003.

L. Steels. The symbol grounding problem has been solved. so what's next? In M. de Vega, editor, *Symbols and Embodiment: Debates on Meaning and Cognition*, chapter 12. Oxford University Press, Oxford, 2008.

L. Steels and F. Kaplan. Aibo's first words: The social learning of language and meaning. *Evolution of Communication*, 4(1):3–32, 2002.

K. Sugiura and N. Iwahashi. Learning object-manipulation verbs for human-robot communication. In *WMISI'07 Proceesdings of the 2007 workshop on Multimodal Interfaces on Semantic Interaction*, pages 32–38. ACM Press, 2007.

N. A. Syed, H. Liu, S. Huan, L. Kah, and K. Sung. Handling concept drifts in incremental learning with support vector machines. In *In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99*, pages 317–321. ACM Press, 1999.

S. Takamuku, Y. Takahashi, and M. Asada. Lexicon acquisition based on object-oriented behavior learning. *Advanced Robotics*, 20(10):1127–1145, 2006.

L. Talmy. *Toward a cognitive semantics: Concept structuring systems (language, speech and communication).* Cambridge, MA: The MIT Press, 2000.

A. Tavakkoli, A. Ambardekar, M. Nicolescu, and S. Louis. A genetic approach to training support vector data descriptors for background modeling in video data. In *Advances in Visual Computing*, pages 318–327. Springer, 2007.

D. M. J. Tax. *One Class Classification: Concept learning in the absence of counter-examples.* PhD thesis, Technische Universiteit Delft, The Netherlands, 2001.

A. L. Thomaz and C. Breazeal. Robot learning via socially guided exploration. In *ICDL2006*, 2006.

A. L. Thomaz and C. Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence Journal*, 172:716–737, 2008.

S. Thrun. *Explanation-based Neural Network Learning: A Lifelong Learning Approach.* Boston, MA: Kluwer, 1996.

M. Tomasello. The cultural origins of human cognition. *Cognitive Development*, 10:131–156, 2000a.

M. Tomasello. Do young children have adult syntactic competence. *Cognition*, 7:209–253, 2000b.

M. Tomasello. *Constructing a lanaguge: a usage based theory of lanaguge acquisition.* Cambridge, MA: Harvard University Press, 2003.

J. M. Toro, M. Nespor, J. Mehler, and L. L. Bonati. Finding words and rules in speech stream: functional differences between vowels and consonants. *Psychological Science*, 19 (2):137–144, 2008.

A. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

M. Uray, P. M. Roth, and H. Bischof. Efficient classification for large-scale problems by multiple lda subspaces. In A. Ranchordas and H. Arajo, editors, *VISAPP (1)*, pages 299–306. INSTICC Press, 2009. ISBN 978-989-8111-69-2.

V. Vapnik. *The nature of statistical learning theory.* New York: Springer-Verlag, 1995.

G. Vigliocco, L. Meteyard, M. Andrews, and K. S. Toward a theory of semantic representation. *Language and Cognition*, 1(2):219–247, 2009.

P. Vogt and H. Coumans. Investigating social interaction strategies for bootstrapping lexicon development. *J. Artificial Societies and Social Simulation*, 6(1), 2003.

Q. H. Wang, L. Seabra Lopes, and D. M. J. Tax. Visual object recognition through one-class learning. In *International Conference on Image Analysis and Recognition (ICIAR 2004)*, Part 1, LNCS 3211, pages 463–469. Springer, 2004.

G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22:418–435, 1992.

C. Yang. *Knowledge and Learning in Natural Language.* Oxford University Press, 2002.

J. Yang, K. Yu, Y. Gong, and T. S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801. IEEE, 2009. ISBN 978-1-4244-3992-8.

H. Yoshida and L. B. Smith. Linguistic cues enhance the learning of perceptual cues. *Psychological Science*, 16(2):90–95, 2005.

C. Yu. The emergence of links between lexical acquisition and object categorization: A computational study. *Connection Science*, 17(3):381–397, 2005.

C. Yu and D. H. Ballard. A multimodal learning interface for grounding spoken language in sensory perceptions. *ACM Transactions on applied perception*, 1:57–80, 2004.

C. Yu and D. H. Ballard. A unified model of early word learning: Integrating statistical and social cues. *Neurocomputing*, 70(13-15):2149–2165, 2007.

D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37:1–19, 1 2004.