

EXPERIENCE-BASED LANGUAGE ACQUISITION:
A COMPUTATIONAL MODEL OF
HUMAN LANGUAGE ACQUISITION

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Brian Edward Pangburn
B.S., Tulane University, 1994
M.S., Louisiana State University, 1999
December, 2002

©Copyright 2002
Brian Edward Pangburn
All rights reserved

To Jack and Jeremiah

Modeling, it should be clear, is an art form.
It depends on the experience and taste of the modeler.
—John Holland, *Hidden Order*

ACKNOWLEDGEMENTS

I would like to begin by thanking my wife, Jaimee, for standing behind me and for believing that I would one day finish my dissertation. Without her sacrifice, this dissertation would not be. I would like to thank my family for instilling in me the importance of a good education and express my gratitude to everyone at The Pangburn Company for being flexible and allowing me the time that I needed for research. I would also like to thank my committee for their patience as life, time and again, interrupted my research.

The material in this work on language acquisition in children is a direct result of my firsthand exposure to the brilliance of both Emily Smith and Jan Norris. Many of the technical roadblocks in the EBLA system were overcome thanks to the insights and friendship of Chris Branton and Carey Patin. The readability of this document owes much to the grammatical prowess of Sara Aguiard, who has always had the patience to proofread my work. Finally, there is no way that this research would have been finished in time without the tremendous help of Jon Ayo, who aided me with the EDISON port and did much of the filming and editing on the seemingly endless pile of videos for EBLA.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	v
LIST OF TABLES.....	ix
LIST OF FIGURES.....	x
ABSTRACT.....	xii
CHAPTER 1 – INTRODUCTION.....	1
1.1 – A Problem of Overwhelming Difficulty.....	1
1.2 – A Partial Solution.....	2
1.3 – What Lies Beneath.....	5
CHAPTER 2 – EARLY LANGUAGE ACQUISITION IN CHILDREN.....	8
2.1 – Introduction.....	8
2.2 – Why Can’t We All Just Get Along?.....	8
2.2.1 – Nature.....	8
2.2.2 – Nurture.....	9
2.2.3 – Epigenesis.....	10
2.2.4 – Everything in Moderation.....	10
2.3 – Experiential Model.....	13
2.4 – Situational-Discourse-Semantic (SDS) Model.....	16
2.5 – Chronology of Development.....	18
2.5.1 – Prenatal Development.....	19
2.5.2 – Birth to One Month.....	19
2.5.3 – One to Four Months.....	20
2.5.4 – Three to Eight Months.....	21
2.5.5 – Seven to Ten Months.....	23
2.5.6 – Ten to Fourteen Months.....	24
2.5.7 – Fifteen Months and Beyond.....	25
2.6 – Summary.....	27
CHAPTER 3 – COMPUTATIONAL MODELS OF PERCEPTUAL GROUNDING AND LANGUAGE ACQUISITION.....	28
3.1 – Introduction.....	28
3.2 – Cross-Situational Techniques for Lexical Acquisition.....	28
3.3 – Force Dynamics and Event Logic for Grounded Event Recognition.....	31
3.4 – X-Schemas, F-Structs, and Model-Merging for Verb Learning.....	32
3.5 – Cross-Modal Acquisition of Shape and Color Words.....	34
3.6 – Acquisition of Words and Grammar for Spatial Description Tasks.....	36
3.7 – Social Learning of Language and Meaning.....	36
3.8 – Summary.....	37

CHAPTER 4 – OVERVIEW OF THE EXPERIENCE-BASED LANGUAGE ACQUISITION MODEL	39
4.1 – Introduction.....	39
4.2 – Developmental Basis for Model	40
4.3 – Computational Basis for Model.....	42
4.4 – Model Abstractions and Constraints.....	42
4.5 – Experiences Processed by the EBLA Model.....	43
4.6 – Entity Recognition.....	44
4.7 – Lexical Acquisition.....	46
4.8 – Summary.....	49
 CHAPTER 5 – IMPLEMENTATION OF THE EBLA MODEL	 51
5.1 – Introduction.....	51
5.2 – Technologies Involved.....	51
5.3 – EBLA Executable Class.....	53
5.4 – Database Connection	54
5.5 – Parameters.....	55
5.6 – Video Processing	57
5.7 – Frame Processing.....	58
5.7.1 – Edge Detection.....	58
5.7.2 – Image Segmentation.....	59
5.7.3 – Polygon Analysis	63
5.7.4 – Intermediate Results.....	64
5.8 – Entity Extraction.....	65
5.8.1 – Object Entities.....	67
5.8.2 – Relation Entities.....	68
5.9 – Lexical Resolution.....	69
5.9.1 – Parsing.....	70
5.9.2 – Cross-Situational Learning	70
5.10 – Summary.....	74
 CHAPTER 6 – EVALUATION OF THE EBLA MODEL.....	 75
6.1 – Evaluation Criteria.....	75
6.2 – Creation of Animations.....	75
6.3 – Evaluation of Animations	76
6.4 – Creation of Videos.....	79
6.5 – Evaluation of Videos	80
6.6 – Summary.....	88
 CHAPTER 7 – FUTURE WORK AND DISSEMINATION.....	 89
7.1 – Introduction.....	89
7.2 – More Complex Experiences.....	89
7.3 – Compound Entities	90
7.4 – Model-Merging.....	90
7.5 – Dynamic Attributes.....	91

7.6 – Graphical User Interface	91
7.7 – Speech Recognition and Speech Synthesis.....	91
7.8 – Syntax	92
7.9 – Virtual Vision System/Game Engine.....	92
7.10 – Feedback	93
7.11 – Knowledge Base	93
7.12 – Emotion, Motivation, and Goals.....	94
7.13 – Dissemination	94
7.14 – Summary	95
CHAPTER 8 – CONCLUSIONS	96
8.1 – A Step in the Right Direction	96
8.2 – How Does EBLA Compare?.....	97
8.3 – Applications	100
8.4 – Summary	100
REFERENCES	102
APPENDIX A – LISTING OF EXPERIENCES PROCESSED BY EBLA	107
APPENDIX B – LISTING OF RESOURCES FOR THE EBLA PROJECT	115
APPENDIX C – SAMPLE JAVADOC FOR EBLA	116
APPENDIX D – SQL USED TO CONSTRUCT THE EBLA DATABASE	120
VITA.....	127

LIST OF TABLES

Table 1. Comparison of Existing Computational Models	38
Table 2. Entity Attributes Calculated by EBLA	45
Table 3. Top-Level Classes Instantiated by EBLA	54
Table 4. EBLA Parameters	56
Table 5. Relation Attributes for EBLA.....	69
Table 6. Animation Description Results When Describing One of Eight Experiences.....	79
Table 7. Accuracy of Video Descriptions.....	86
Table 8. Sample Video Description Results for Ten of 167 Video Experiences.....	87
Table 9. Comparison of EBLA with Other Computational Models	98

LIST OF FIGURES

Figure 1. Frames from an Experience Processed by EBLA	3
Figure 2. Frames Following the Detection of Significant Objects	3
Figure 3. Referential Ambiguity Faced by EBLA.....	4
Figure 4. Abstractions for Computational Model.....	5
Figure 5. Relationships among the Representational Levels in the Experiential Model.....	14
Figure 6. Situational Continuum.....	16
Figure 7. Discourse Continuum.....	17
Figure 8. Semantic Continuum	17
Figure 9. Integrated SDS Model.....	18
Figure 10. SLIDE X-Schema.....	33
Figure 11. Method Used by EBLA to Process Experiences	40
Figure 12. Frames from Various Animations Processed by EBLA.....	44
Figure 13. Frames from Various Videos Processed by EBLA	44
Figure 14. General Architecture of EBLA Software System	52
Figure 15. EBLA Database Tables and Relationships.....	53
Figure 16. Video Frames and Polygon Tracings from Original Edge Detector	59
Figure 17. Normal Segmentation, Oversegmentation, and Undersegmentation	61
Figure 18. Sample Segmented Images.....	65
Figure 19. Sample Polygon Images	65
Figure 20. Set Intersection Query for Lexical Resolution.....	72
Figure 21. First Set Difference Query for Lexical Resolution.....	73
Figure 22. Second Set Difference Query for Lexical Resolution	73

Figure 23. Variation in Shape of Arm/Hand Used in Animations.....	76
Figure 24. Average Lexical Acquisition Time for Animations	77
Figure 25. Lexeme Mapping Success Rates for Different Minimum Standard Deviations	78
Figure 26. Stage Used to Film EBLA Experiences	80
Figure 27. Polygon Traces from a Single Video Demonstrating Normal Segmentation, Undersegmentation, and Oversegmentation	82
Figure 28. Average Lexical Acquisition Time for Videos	84
Figure 29. Lexeme Mapping Success Rates for Different Minimum Standard Deviations	85
Figure 30. Distribution of Correct, Incorrect, and Unknown Lexemes in Video Descriptions...	87

ABSTRACT

Almost from the very beginning of the digital age, people have sought better ways to communicate with computers. This research investigates how computers might be enabled to *understand* natural language in a more humanlike way. Based, in part, on cognitive development in infants, we introduce an open computational framework for visual perception *and* grounded language acquisition called Experience-Based Language Acquisition (EBLA). EBLA can “watch” a series of short videos and acquire a simple language of nouns and verbs corresponding to the objects and object-object relations in those videos. Upon acquiring this *protolanguage*, EBLA can perform basic scene analysis to generate descriptions of novel videos.

The general architecture of EBLA is comprised of three stages: vision processing, entity extraction, and lexical resolution. In the vision processing stage, EBLA processes the individual frames in short videos, using a variation of the mean shift analysis image segmentation algorithm to identify and store information about significant objects. In the entity extraction stage, EBLA abstracts information about the significant objects in each video and the relationships among those objects into internal representations called *entities*. Finally, in the lexical acquisition stage, EBLA extracts the individual lexemes (words) from simple descriptions of each video and attempts to generate entity-lexeme mappings using an inference technique called cross-situational learning. EBLA is not primed with a base lexicon, so it faces the task of bootstrapping its lexicon from scratch.

The performance of EBLA has been evaluated based on acquisition speed and accuracy of scene descriptions. For a test set of simple animations, EBLA had average acquisition success rates as high as 100% and average description success rates as high as 96.7%. For a larger set of real videos, EBLA had average acquisition success rates as high as 95.8% and average

description success rates as high as 65.3%. The lower description success rate for the videos is attributed to the wide variance in entities across the videos.

While there have been several systems capable of learning object or event labels for videos, EBLA is the first known system to acquire both nouns *and* verbs using a grounded computer vision system.

CHAPTER 1 – INTRODUCTION

1.1 – A Problem of Overwhelming Difficulty

In a recent book about HAL, the computer in the movie *2001: A Space Odyssey*, David G. Stork wrote:

Imagine, for example, a computer that could look at an arbitrary scene—anything from a sunset over a fishing village to Grand Central Station at rush hour—and produce a verbal description. This is a problem of overwhelming difficulty, relying as it does on finding solutions to both vision and language and then integrating them. I suspect that scene analysis will be one of the last cognitive tasks to be performed well by computers. (Stork 2000, 8)

Unfortunately, true humanlike scene analysis is even more difficult than Stork indicates. This is because the solution to the language problem may very well depend on the solution to the vision problem, and on the broader problem of perception in general. Sensory perception gives meaning to much of human language, and to convey such meaning to a computer may require that perception be integrated with language from the very start.

The goal of this research is to construct a simplified version of the dynamic scene analysis system described by Stork (Stork 2000) and to investigate how computers might be enabled to understand language in more humanlike terms. While traditional, top-down research fields such as natural language processing (NLP), computational linguistics, and speech recognition and synthesis have made great progress in allowing computers to *process* natural language, they typically do not address *perceptual understanding*. In these fields, meaning and context for a given word are based solely on other words and the logical relationships among them.

To make this clearer, consider the following Webster’s definition of *apple*: “The fleshy usually rounded and red or yellow edible pome fruit of a tree of the rose family.” (Webster’s 1989) Using traditional approaches, a computer might be able to determine from such a

definition that an apple is “edible,” that it is a “fruit,” and that it is usually “rounded and red or yellow.” But what does it *mean* to be “rounded and red”? People understand these words because their conceptual representations are grounded in their perceptual experiences. As for more abstract words, many have perceptual analogs or can be defined in terms of grounded words. Although it is unlikely that any two people share identical representations of a given word, there are generally enough similarities for that word to convey meaning. If computers can be enabled to ground language in perception, ultimately communication between man and machine may be facilitated.

1.2 – A Partial Solution

This research investigates the challenges of cognitive development and language acquisition for both children and computers. It details a new software framework, Experience-Based Language Acquisition (EBLA), that acquires a childlike language known as protolanguage in a bottom-up fashion based on visually perceived experiences. EBLA uses an integrated computer vision system to watch short videos and to generate internal representations of both the objects and the object-object relations in those videos. It then performs language acquisition by resolving these internal representations to the individual words in protolanguage descriptions of each video. Upon acquiring this grounded protolanguage, EBLA can perform basic scene analysis to generate simplistic descriptions of what it “sees.”

EBLA operates in three primary stages: vision processing, entity extraction, and lexical resolution. In the vision processing stage, EBLA is presented with *experiences* in the form of short videos, each containing a simple event such as a hand picking up a ball. EBLA processes the individual frames in the videos to identify and store information about significant objects. In the entity extraction stage, EBLA aggregates the information from the video processing stage

into internal representations called *entities*. Entities are defined for both the significant objects in each experience and for the relationships among those objects. Finally, in the lexical acquisition stage, EBLA attempts to acquire language for the entities extracted in the second stage using protolanguage descriptions of each event. It extracts the individual lexemes (words) from each description and then attempts to generate entity-lexeme mappings using an inference technique called cross-situational learning. EBLA is not primed with a base lexicon, so it faces the task of bootstrapping its lexicon from scratch.

For example, assume EBLA is presented with a short video of a hand picking up a ball and the protolanguage description “hand pickup ball.” In the video processing phase, EBLA would extract the individual frames from the movie (see figure 1), and determine the location of the significant objects in each (see figure 2).

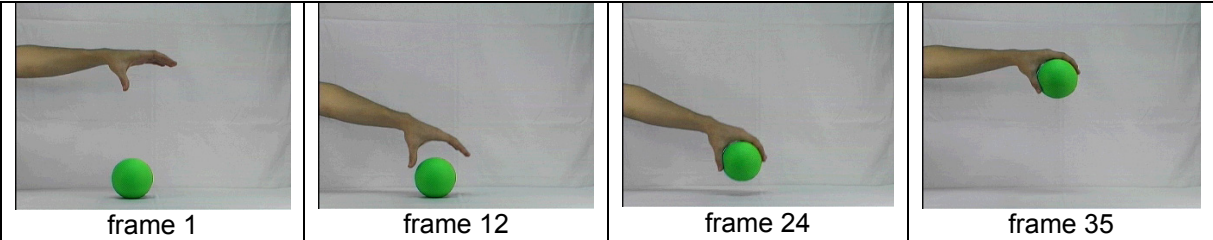


Figure 1. Frames from an Experience Processed by EBLA

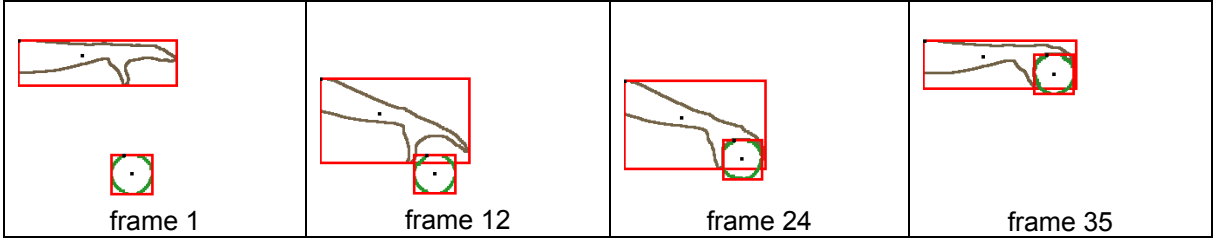


Figure 2. Frames Following the Detection of Significant Objects

In the entity extraction phase, EBLA would analyze the information in all of the frames to establish object entity definitions for the hand and the ball, and a relation entity definition for the

spatial relationship between the hand and the ball. In the lexical resolution stage, EBLA would attempt to resolve the lexemes “hand,” “pickup,” and “ball” to their respective entities.

Since EBLA is not primed with any entities, lexemes, or mappings, it faces ambiguity in its early experiences (see figure 3). If the above example were its first experience, it would have no way to establish any of the entity-lexeme mappings. In order to overcome this, EBLA compares both entities and lexemes across multiple experiences to resolve ambiguity. A more detailed discussion of this process is presented in section 4.7.

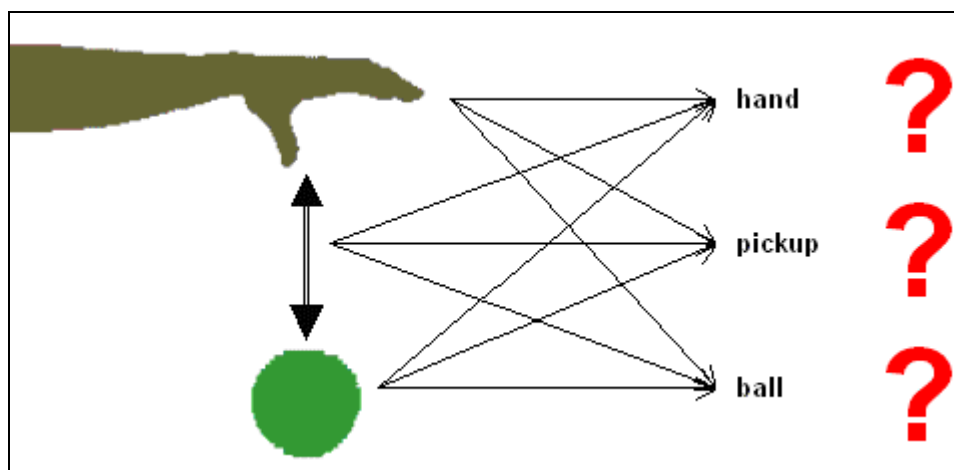


Figure 3. Referential Ambiguity Faced by EBLA

In order to implement the EBLA project in a reasonable amount of time using available technologies, the model has been constrained in several ways. First, EBLA’s perceptual system is limited to a two-dimensional computer vision system. Second, the protolanguage descriptions delivered to EBLA are in textual form. A graphical representation of these first two abstractions is provided in figure 4. The third way that EBLA has been constrained is that it only attempts to acquire an unstructured protolanguage of nouns and verbs. EBLA cannot resolve other parts of speech such as adjectives or adverbs, and it makes no attempt to incorporate syntax. Finally, EBLA operates in an unsupervised manner in that it does not get any feedback on its performance. Hopefully, all of these constraints present a worst-case scenario, and by adding

additional perceptual capabilities, language structure, or a feedback system, its performance could be improved. A more detailed description of the project constraints is presented in section 4.4.

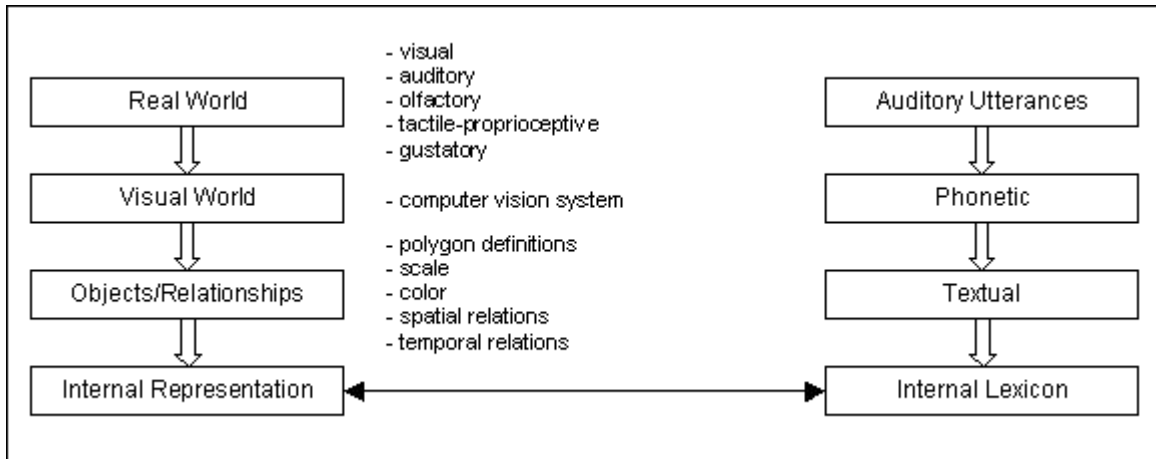


Figure 4. Abstractions for Computational Model

In order to facilitate the future elimination of some of the constraints placed on the EBLA Model, it has been designed as an open framework with expansion and extension in mind. For example, it should be a fairly straightforward process to modify EBLA to accommodate a speech recognition or a speech synthesis module. EBLA has been coded entirely in Java and documented following the JavaDoc conventions. The open-source PostgreSQL database system is used for all of EBLA's data storage and retrieval. EBLA has been developed using open-source and/or freely available tools whenever possible and has been successfully tested on both the Windows and Linux platforms. A listing of available resources for EBLA is provided in appendix B.

1.3 – What Lies Beneath

Chapter 2 investigates current theories of cognitive development and lexical acquisition in infants and toddlers in order to establish some developmental basis for EBLA. First, an overview of the nature versus nurture debate is provided, focusing on research that lies in the

middle and attempts to bridge the two philosophies. Next, two integrated models of development are presented: the Experiential Model formulated by Katherine Nelson, a Professor of Developmental Psychology at City University of New York, and the Situational-Discourse-Semantic (SDS) Model developed by Janet Norris and Paul Hoffman, Professors of Communication Sciences and Disorders at Louisiana State University. Chapter 2 concludes with a chronological outline of cognitive and linguistic development through the first year-and-a-half of life based on the Experiential Model and the SDS Model as well as other pertinent research.

Chapter 3 provides a summary of several bottom-up computational models of grounded perception and language acquisition related to EBLA. These include a system that acquires word-to-meaning mappings for conceptual symbols; several related systems that perform vision-based event recognition; a system that acquires verbs using a virtual proprioceptive model; a system that acquires color and shape words using a vision system and recorded audio; a system that acquires object names, spatial terms, and syntax for computer generated images of various rectangles; and finally, a system that performs vision-based acquisition of object names based on social mediation.

Chapter 4 introduces the EBLA Model in general terms. It begins with some remarks about how the model relates to existing developmental and computational research. Next, the abstractions and constraints used for EBLA are outlined, and the experiences used to evaluate the model are discussed. The chapter concludes with an overview of the entity recognition and lexical acquisition mechanisms employed by EBLA.

Chapter 5 discusses the technology behind EBLA and details the model's implementation on a module-by-module level. Chapter 6 details the evaluation of EBLA including the methods used, the data sets involved, and the results obtained. Chapter 7 summaries long-term and short-

term goals for further research and outlines the dissemination plans for EBLA. Finally, chapter 8 summarizes this research with a discussion about the significance of the results, a comparison with other research, and remarks on possible applications.

CHAPTER 2 – EARLY LANGUAGE ACQUISITION IN CHILDREN

2.1 – Introduction

This chapter surveys a variety of theories of early cognitive development and language acquisition in children. A better understanding of the processes involved in child development can provide much insight into how a computational model might accomplish the same tasks. This is not to say that the model detailed in this work is a simulation of human processes, but rather that the child and computer face analogous hurdles.

Research on child development involves a wide variety of studies including cognitive science, developmental psychology, communication disorders, linguistics, biology, and genetics. Unfortunately, there is no single unifying theory agreed to by each of the involved domains of research. Since the task of this research is *modeling*, it will only be necessary to extract common features among the theories that lend themselves to the development of a bottom-up computational model.

2.2 – Why Can't We All Just Get Along?

As an outsider to the field, it can be rather confusing to study the sciences behind child development. While the processes by which a newborn develops the skills to function as a member of society are fascinating, they are the subject of much debate. Much of this debate seems unnecessary as the various camps on development have more in common than one would be led to believe. At the root of the problem is the age-old argument of nature versus nurture. In the next few sections, some of the more common theories of development are discussed.

2.2.1 – Nature

The nature-centric view of development is known as nativism and focuses on functions and behaviors that are innate. Generally members of this camp believe that infants are born with

innate, domain-specific cognitive structure to handle specialized functions including face recognition, language processing, and mathematics. Learning is little more than a process of fine-tuning this cognitive structure for a particular environment. For example, grammar, is thought to have universal principles that underlie all languages. As children are exposed to their native language, they tune parameters for that language, determining things such as verb placement in phrases. For further discussion of nativism, see Elman, et. al. (Elman, et. al.1999) and Pinker (Pinker 2000).

A variation of the nativist view of development is the evolutionist view. While some in the nativist camp believe that innate cognitive structures come into existence spontaneously as coherent functional units, in the evolutionist view, all innate function is seen as a direct result of Darwinian biological evolution. Language and other brain functions are thought to have evolved slowly over time, thus appearing in some crude form in the evolutionary ancestors of man. The distinction between the nativist and evolutionist views of development dates back to a conflict between Charles Darwin and linguist Max Muller in the late 1800's. Muller took the stance that language is one of the major traits separating humans from the rest of the animals and, therefore, could not have evolved from some related function in lower animals. Unfortunately, about a century later, modern linguist Noam Chomsky took a seemingly related stance regarding language that perpetuated the divide between the nativist and evolutionist views.

His combination of an insistence on the biological nature of language with a refusal to look at the origins of that nature—and his blanket statements about the futility of any such enterprise—turned off many in the evolutionary community who might otherwise have been supportive. (Calvin and Bickerton 2001, 198)

2.2.2 – Nurture

The nurture-centric view of development is known as empiricism and focuses on environmental effects on learning. Generally, members of this camp believe that infants are born

with only domain-general cognitive structure and learn specialized functions based on environmental stimuli. Infants are thought to be born without any task-specific modules. “Learning, in this view, involves a copying or internalizing of behaviors which are present in the environment.” (Elman, et. al. 1999, 1) The behaviorism movement in psychology, which focuses on stimulus-response mechanisms as a basis for learning, is one of the more well-known empiricist approaches.

2.2.3 – Epigenesis

Many of the more modern theories of development are based on epigenesis, a *combination* of nature and nurture. The roots of epigenesis lie in the works of Piaget and other classic developmentalists during the first half of the twentieth century. They began to study both genetic *and* environmental factors as the path to cognition. (Nelson 1998) Generally, members of the epigenetic camp believe that domain-specific cognitive structure *emerges* from the *interactions* between domain-general cognitive structure and experience.

A more recent technical spin on epigenesis is the popular connectionist view of development. It combines discoveries about the workings of the brain with modern computational techniques to model various cognitive processes. (Elman, et. al. 1999)

As extensions to the epigenetic viewpoint, several newer models have integrated the specific impact of social and cultural mediation on child development. Two of these models actually form the developmental basis for this research and are discussed in detail later in this chapter.

2.2.4 – Everything in Moderation

The problem with much of the literature on language and development is that researchers far too often entrench themselves in one camp or another and then quote the competition out of

context to prove a point. Linguists are portrayed as studying language in a vacuum, naively believing in a magical “language organ” or “grammar gene.” Connectionists are portrayed as oversimplifying language and building toy models that only achieve limited results. All of the camps are stereotyped, and these stereotypes are often based on antiquated themes.

Fortunately, several recent works have finally undertaken the tasks of dispelling myths and attempting to reconcile the disparate camps on language and development. These works bring Chomsky back in line with Darwin, and demonstrate how a connectionist network might produce innate function if genetics control much of the wiring. (Calvin and Bickerton 2001; Pinker 2000) The truth is that no one yet completely understands the brain, language, *or* child development. Prejudices aside, most modern researchers can gain insights from certain principles of nativism/evolutionism, empiricism, *and* epigenesis.

Evolution today implies a lot more than it did in the days of Darwin and Muller. Natural selection is only part of the picture. Genetics have shown that while certain genes can be tied to specific traits, it is the complex interaction among large sets of genes that make humans *uniquely human*. Concepts such as autocatalytic sets and complexity theory have provided plausible explanations for nonlinear, emergent behavior in evolution. (Kauffman 1993; 1995; Waldrop 1993) As researchers continue to discover more about the human genome, there is no doubt that science will reveal what is and is not innate. Recently, in fact, it was discovered that mutations in the FOXP2 gene about 200,000 years ago may have given humans the capacity for speech.

A mounting body of research suggests that the mutant gene conferred on human ancestors a finer degree of control over muscles of the mouth and throat, possibly giving those ancestors a rich new palette of sounds that could serve as the foundation of language. (Gillis 2002)

While there are few pure empiricists in modern times, one cannot simply dismiss the fact that there are a lot of environment-dependent concepts that humans learn. The world changes far

too quickly for many types of behavior and function to be innate. Knowledge of how to program a personal computer, for example, is in no way innate. Humans must adapt to an ever-changing world using a skill set quite different from that of their ancestors.

At first glance, epigenesis seems to achieve a happy medium between nature and nurture, but studying the interactions of genetics and environment is a big undertaking and has a long way to go. Connectionism seems to be a promising avenue, but if not applied carefully, it can easily be reclassified as a form of empiricism. The neural networks most commonly used to model connectionism are powerful tools capable of learning by capturing complex, nonlinear relationships among stimuli, but they generally learn from experience. Sometimes the only innate, “genetic” components of a neural network are the underlying learning algorithm and the assumptions and constraints on the model. (Elman, et. al. 1999)

For all three camps, the biggest debate seems to be the extent to which cognitive function, and in particular, language, is innate. Modern linguists such as Pinker seem to believe that there are innate circuits for language in the brain, but that there is no single “language organ.”

The developing brain may take advantage of the disembodied nature of computation to position language circuits with some degree of flexibility. Say a variety of brain areas have the potential to grow the precise wiring diagrams for language components. An initial bias causes the circuits to be laid down in their typical sites; the alternative sites are then suppressed. But if those first sites get damaged within a certain critical period, the circuits can grow elsewhere. (Pinker 2000, 323)

In contrast to this, the connectionists have shown that some language components such as rules of grammar can be learned simply from exposure to a training set of examples. (MacWhinney 1998) While such results are impressive and seem to demystify language to a certain extent, why should every child relearn all aspects of language from scratch? It seems plausible that the

brain may be genetically encoded with powerful pattern analysis circuitry, more complex and specific than connectionist networks, but not “wired” specifically to perform cognitive tasks such as detecting underlying principles of language.

2.3 – Experiential Model

Katherine Nelson (Nelson 1998) has worked to bring together many of the domains involved in the cognitive development of children with special emphasis on the role played by language. She views language and cognition as heavily intertwined—language cannot develop without early, nonlinguistic cognitive function, and full cognitive development cannot occur without language. Nelson takes an *experiential* approach to her work, focusing on how children adapt to meet their current needs and how that adaptation then affects their future experiences.

Nelson’s Experiential Model is centered on *events* in the child’s environment rather than *objects*. Nelson broadly defines an event as “an organized sequence of actions through time and space that has a perceived goal or end point.” (Nelson 1998, 93-94) Events place objects and actions on those objects in the context of their ultimate goal or purpose, adding temporal ordering with a beginning and an ending. A child’s perception, processing, classification, and storage of events form his/her *mental event representations* (MERs). The MER becomes the cognitive building block for increasingly complex knowledge representation and, ultimately, natural language.

The Experiential Model places cognitive development in the context of a social and cultural environment. Nelson focuses “on the emergence in development of language as a representational system, both for internal cognitive functions and for external communicative functions.” (Nelson 1998, 12) She envisions development progressing through episodic, mimetic, and linguistic stages of representational ability. Each new representational ability

builds on the last and is incorporated into a child's MERs. The adult is said to have a hybrid mind that is composed of all three types of representations. Figure 5 is taken from Smith (Smith 1999) and diagrams the relationships among the representations in the Experiential Model.

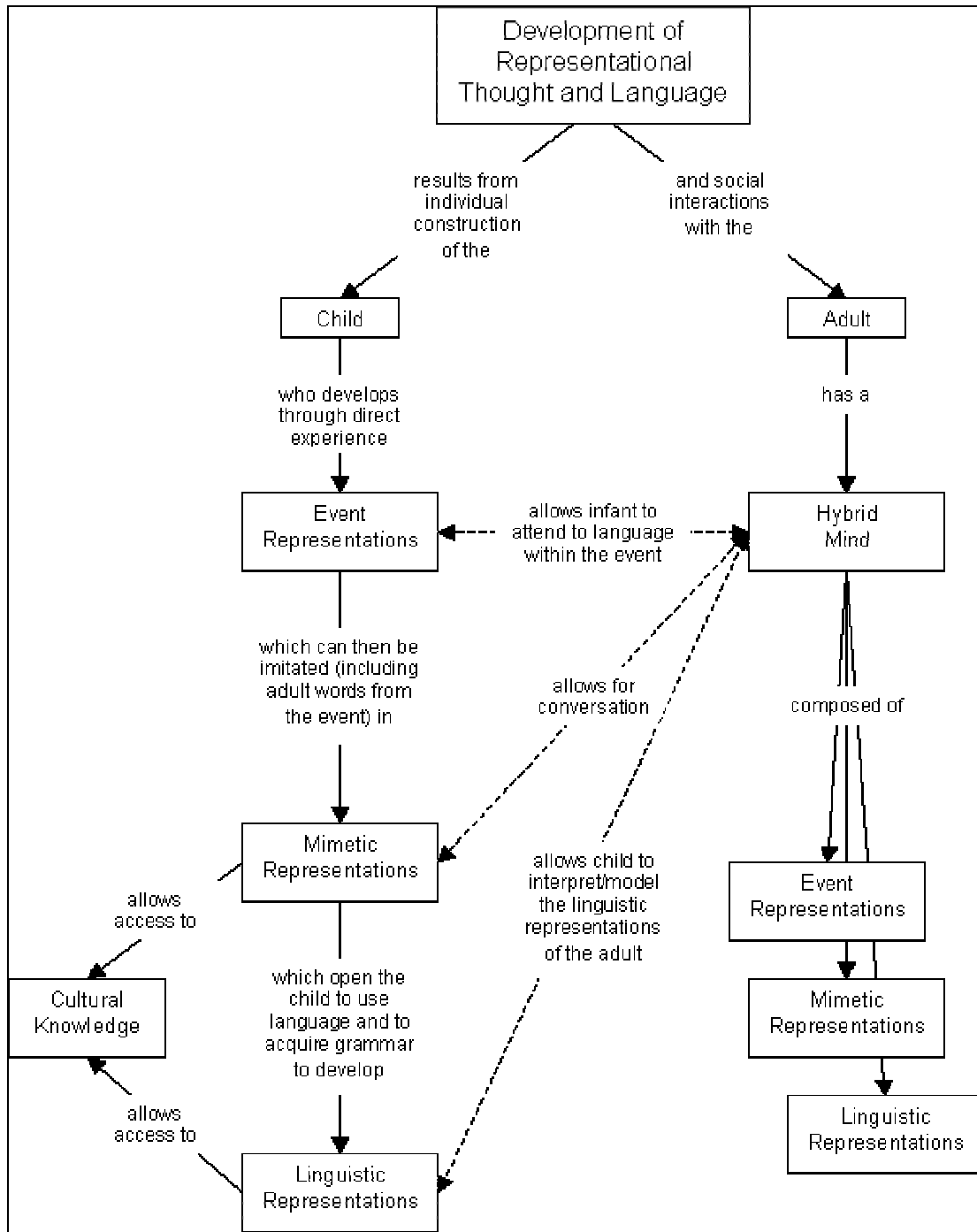


Figure 5. Relationships among the Representational Levels in the Experiential Model (Smith 1999, 16)

Episodic representation involves the perception and storage of a pattern of stimuli as a unit. This lowest level of event representation can only be recalled in the presence of similar stimuli and cannot be reflected upon in the absence of the stimuli that triggered it. Language at this stage is simply auditory stimuli incorporated into the representation of an event.

Mimetic representation involves the intentional imitation of some behavior. This level of event representation allows for recall in the absence of triggering stimuli. Its primary uses are communication of events and practice of skills. Language, at this stage, involves words uttered during the communicative reenactment of an event or the practice of motor skills associated with vocalization. Words, at this stage, only have meaning as part of an event representation. They do not have stand-alone semantic value.

Linguistic representation involves the use of language as a system of meaning. Initially, linguistic representation can only be used to communicate first-hand experiences, but eventually it emerges to allow communication and understanding of third-party experiences. As linguistic representation develops, both existing MERs and new experiences are reorganized in terms of language. Language allows for the full cultural mediation of experiences by others as it becomes possible to share abstract internal representations. According to Smith,

Thinking in language is coming to think culturally instead of only thinking individually or even socially. Language does not belong to just the individual or the family, its forms and structures are culturally embedded. (Smith 1999, 20)

Once linguistic representation emerges, it becomes possible to utilize external symbolic storage (ESS). ESS is a sort of external memory that can aid in symbolic processing. It allows information to be processed in nonlinear chunks and to persist beyond the lifetime of any single individual. Written language, mathematics, science, and history are just a few of the knowledge domains made possible by ESS. The emergence of ESS in socio-cultural history generated an

explosion in the cultural discovery, representation, and storage of knowledge that continues to this day. (Nelson 1998; Smith 1999)

2.4 – Situational-Discourse-Semantic (SDS) Model

Norris and Hoffman (Norris and Hoffman 2002) developed the Situational-Discourse-Semantic (SDS) Model as an integrated view of child development for use in diagnosing and treating communication disorders. The SDS Model tracks child development along situational, discourse, and semantic continua.

The situational continuum (see figure 6) tracks a child’s capacity to *represent* information as it is displaced spatially, temporally, and logically from external perception. The discourse continuum (see figure 7) tracks a child’s capacity to *organize* his/her internal representations. Finally, the semantic continuum (see figure 8) tracks a child’s capacity to *process* his/her internal representations and to associate meaning with his/her organized knowledge.

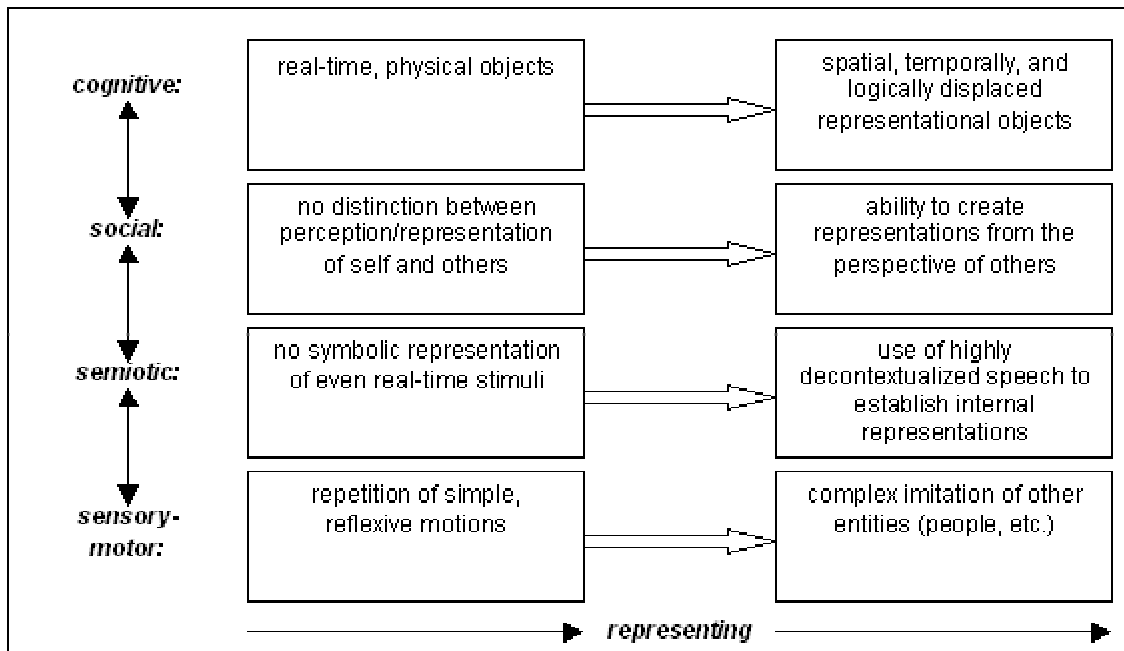


Figure 6. Situational Continuum

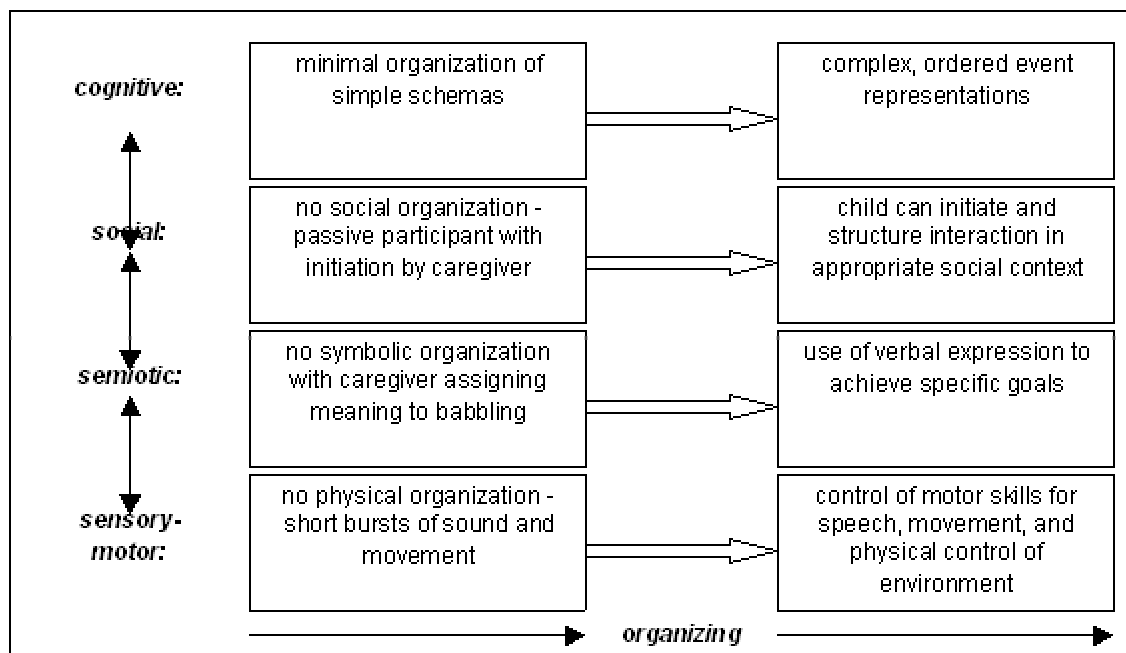


Figure 7. Discourse Continuum

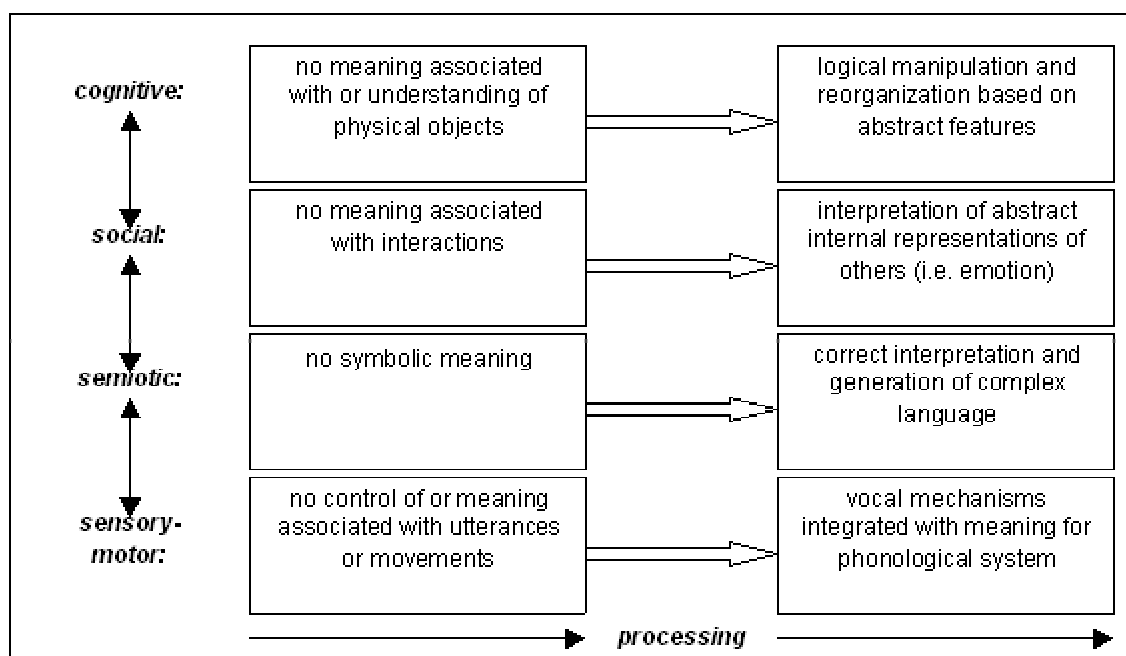


Figure 8. Semantic Continuum

The three SDS continua are evaluated in terms of four overlapping knowledge domains: *cognitive*, *social*, *semiotic*, and *sensory-motor*. The cognitive domain includes knowledge of objects, their attributes, and the relationships among them. The social domain includes

knowledge of social and cultural dynamics as well as knowledge of the thoughts, beliefs, and goals of others. The semiotic domain includes knowledge of symbolic representation including gestures, signs, and language. The sensory-motor domain includes knowledge of one's own body and how to use it to interact with one's environment. Figure 9 illustrates that SDS is an integrated model with three heavily interdependent continua evaluated over four overlapping domains.

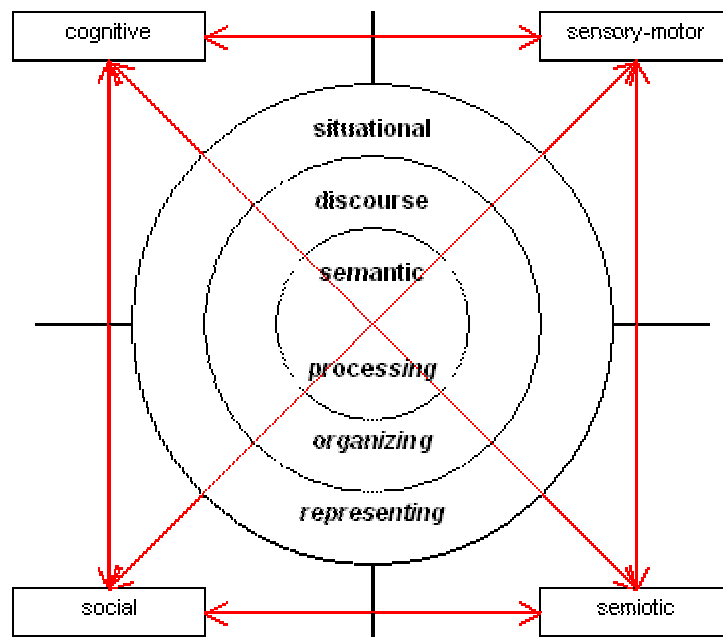


Figure 9. Integrated SDS Model

2.5 – Chronology of Development

The following sections provide a chronological outline of cognitive and linguistic development in infants. The stages of development presented are general trends and do not have absolute boundaries. This chronology is based primarily on Nelson's Experiential Model and Norris and Hoffman's SDS Model, but also draws from several other sources for supporting material.

Note that the terms “mental event representation,” “MER,” “event representation,” and “internal representation” are used somewhat interchangeably in the literature on the Experiential Model and the SDS Model. For this work, “internal representation” will be used to refer to the infant’s early mental representations of disconnected perceptual knowledge. The term “event representation” will be used in lieu of “mental event representation” or “MER” to refer to the infant’s preconceptual mental representations of an entire event. The term “concept” will be reserved for more advanced event representations as the child begins to form categories based on function rather than just on perceptual characteristics. Just as there are no absolute boundaries for the stages of development, internal representations, event representations, and conceptual representations are subjective and overlapping.

2.5.1 – Prenatal Development

It might be a bit surprising to have a chronology of infant development begin prior to birth, but there is evidence that infants begin processing some information prenatally, especially sound.

Infants are exposed to linguistically relevant stimulation while still in the womb, and there is evidence to suggest that infants react to prenatal stimulation in developmentally favorable ways. This evidence includes the finding that neonates can generally distinguish between their own mother’s voice and the voice of another mother, and the observation that newborns seem to prefer the language spoken by their mother to disparate languages. (Locke 1993, 23)

This indicates that some very basic internal representations for auditory information may be established prior to birth. Such representations could aid the newborn in bonding to his/her parents and perceiving his/her environment following birth.

2.5.2 – Birth to One Month

At birth, an infant is thrust into a world full of new and unfamiliar sensations, which he/she must begin to process and make sense of in order to survive. An infant is most likely born

with neither a vacant mind nor predetermined templates for the situations he/she is to encounter. Rather, he/she comes equipped with the basic biological ability to sense and perceive his/her environment as well as the cognitive ability to hone those perceptions and begin to make sense of them.

Aside from any isolated perceptual learning from the womb, for the newborn, all sensory-perceptive information is novel. Until the internal representations begin to form for these novel stimuli, there is no way to recall or reflect on an object after it is taken away. Thus, the infant can only attend to people or objects in his/her immediate presence. Infants are egocentric in that they cannot make distinctions between “self” and “others.”

The first internal representations that do form in the first month of life are heavily grounded in perception and are discrete. The face of a parent, the taste of milk, and the touch of a hand are represented as-is, without meaning or ordering. This also means that something as simple as a bottle may generate different representations when viewed from different angles.

As a final note, vocalizations in the first month of life are primarily limited to cries of hunger, fatigue, pain, etc. The newborn has little control of its vocal facilities. (Norris and Hoffman 2002)

2.5.3 – One to Four Months

During the first few months of life, the infant is repeatedly exposed to a series of fairly standardized events such as feeding, bathing, rocking, and putting to bed. These *event routines* are not identical, but they have many similarities including location, participating caregivers, and objects involved. As the infant is repeatedly exposed to the same objects from various viewpoints, multi-modal sensory information for those objects including appearance, taste, smell, and sound gets merged into his/her internal representations. During this stage, the infant begins

to construct internal representations, not only for perceptual entities, but also for the basic relationships among those entities. Examples might include relating the appearance of a bottle with the taste of milk, or relating the appearance of a crib mobile with the sound of music. Nelson states, “the child is learning or constructing not laws of object relations, but rules of social interaction with objects *as they apply to the infant during that stage of the child’s development.*” (Nelson 1998, 95) As these internal representations mature, event representations start to form for the most repetitive events in the infant’s environment, integrating sequences of objects, their relations, and the social context in which they occur. (Nelson 1998)

The results of the infant’s early cognitive development can be seen in his/her actions. Familiar objects are recognized and attended to, but only while the stimulus is present. The child watches and responds to other people and possesses enough sensory-motor control to repeat his/her own gestures if mimicked and encouraged by a caregiver. Responses to a stimulus become coordinated “so that seeing an object elicits an attempt to touch it, and hearing an object elicits a directional turn toward the source of the sound.” (Norris and Hoffman 2002, 33) The infant also develops the ability to react to stimuli in anticipation. Based on his/her developing representational abilities, he/she can predict actions for familiar routines.

Vocally, the infant develops distinct cries and vocalizations for different situations. Speech sounds are limited to crude syllables and cooing. (Norris and Hoffman 2002)

2.5.4 – Three to Eight Months

During this period, event representations are continuously recombined, incorporating new knowledge and becoming further interconnected. It is important to note that these event representations are comprised of more than just perceptual sensory data. Infants are subjectively influenced by emotion, social interactions, cultural arrangements, etc. In addition, event

representations incorporate basic physical knowledge of the environment. “Infants of 6 months of age have been shown to expect objects to be three-dimensional, to be substantial, to occupy space, to fall to earth when dropped, and to cause other objects to move on impact.” (Nelson 1998, 31)

Event representations also start to incorporate words, but they do not yet hold any particular meaning. Rather, words at this stage are just part of the perceptual information associated with an event. Caregivers enhance these associations by engaging their children with simple language to signal events.

As the internal representations for an object become further abstracted from sensory perception, infants are able to recall and reflect upon them even after the object has been removed. Children begin to observe and represent part-to-whole relationships, and start to recognize differences between themselves and others. Children also start to incorporate cause-effect information into their internal representations of objects. They focus on their own actions more than the actions of others, and credit themselves with causing actions even if a caregiver is involved. This new level of organization allows them to participate in simple interactive games like peek-a-boo.

Representational *meaning* begins to emerge through the formation of precepts. Precepts are simple patterns of form or function that add prototypical categories to internal representations. Note that these early categories are based on the infant’s view of an object’s role within an event and may be unrelated to traditional adult categorizations of those same objects. (Norris and Hoffman 2002)

During this stage of development, infants’ vocalizations grow to include vowel-like sounds and early babbling.

Between five and seven months babies begin to play with sounds, rather than using them to express their physical and emotional states, and their sequences of clicks, hums, glides, trills, hisses, and smacks begin to sound like consonants and vowels. Between seven and eight months, they suddenly begin to babble in real syllables like *ba-ba-ba*, *neh-neh-neh*, and *dee-dee-dee*. The sounds are the same in all languages. (Pinker 2000, 268-9)

Many infants start to say “dada” about seven months, but initially, this is babbling, not a linguistic representation for “father.”

2.5.5 – Seven to Ten Months

During the second six months of life infants begin to exhibit more control over their environments. They begin to actively play with their toys, and as they learn to creep and crawl, they can start to choose where and with what they want to play. In addition, “when infants begin to locomote, they stop defining the physical environment in relation to their own location.” (Locke 1993, 100)

Infants continue to learn their roles in “social” events, which are primarily mediated by their caregivers. Language continues to be incorporated into internal representations and “beginning at about 9 or 10 months, children begin to give evidence of responding to some specific language forms, including names of family members, signals for games (e.g., “patty-cake”) and routines (e.g., “bath”), and so on.” (Nelson 1998, 112) Incorporation of language is greatly affected by the way that caregivers speak to and interact with their children. Voice inflection, slowed speech, repetition of infant vocalizations, and treatment of the child as a “participant” in conversation all help to mediate the process. (Nelson 1998)

At this stage in development, “the focus of the child’s attention is no longer on performing actions using his own body, but rather on effects of those actions on objects. From the general event representation, a script begins to form and generalize across events with similarities.” (Norris and Hoffman 2002, 43) In addition to establishing basic scripts, the child

begins to detect and utilize means-ends relationships in various contexts. The child also begins to imitate familiar cultural gestures such as pointing and waving, but cannot separate those gestures from the events during which they occur. (Norris and Hoffman 2002)

Vocally, babbling continues and multi-syllable utterances (e.g. “daba,” “dade”) may start to emerge.

2.5.6 – Ten to Fourteen Months

This stage in development is quite remarkable because it involves the emergence of both concept formation and language. Concepts arise from preconceptual event representations as children start to form categories based on function rather than just perceptual characteristics. These categories include both thematic and taxonomic knowledge. Thematic categories group objects linked by their connection to an event. For example, “bottle” and “juice” might be thematically categorized under the “snack” event. Taxonomic categories are linked by their substitutability in a slot-filler. For example “cookies” and “bananas” could be interchanged in a slot-filler such as *EAT(x)*. This early conceptual understanding leads to more appropriate use of tools and toys in the child’s activities. (Nelson 1998) The child also develops the ability to represent and find hidden objects. (Norris and Hoffman 2002)

Children’s first concepts generally correspond to the *basic-level categories* described by prototype theory. Prototype theory is a modern theory of categorization, which proposes that:

The internal structure of natural language categories is organized around prototypes; that categories have graded structure, with more central and more peripheral members; and that categories are structured in terms of family resemblances, that is, overlapping features, none of which are either necessary or sufficient. (Nelson 1998, 227)

Within prototype theory, basic-levels exist between superordinates and subordinates in the taxonomic hierarchy. For example, the basic-level category “car” might have the superordinate

“vehicle” and the subordinate “Ford.” Basic-level categories are defined as having the following characteristics:

- The highest level at which category members have similarly perceived overall shapes.
- The highest level at which a single mental image can reflect the entire category.
- The highest level at which a person uses similar motor actions for interacting with category members.
- The level at which subjects are fastest at identifying category members.
- The level with the most commonly used labels for category members.
- The first level named and understood by children.
- The first level to enter the lexicon of a language.
- The level with the shortest primary lexemes.
- The level at which terms are used in neutral contexts. For example, *There's a dog on the porch* can be used in a neutral context, whereas special contexts are needed for *There's a mammal on the porch* or *There's a wire-haired terrier on the porch*.
- The level at which most of our knowledge is organized. (Lakoff 1990, 46)

Of particular interest is the fact that there is a direct correlation between the first categorical structure used by children and the first words acquired by children. (Lakoff 1990, Nelson 1998)

Children's first words usually emerge on or shortly after their first birthday. By fifteen months, children use an average of ten words, however this number can range from one or two to over 100. (Nelson 1998) First words are not language, per se, but rather a form of protolanguage. Protolanguage is basically the simplistic use of words with little or no syntactic structure. Utterances are small and generally do not include articles, prepositions, etc. Word order is not important, and words can even be omitted as the speaker desires. (Calvin and Bickerton 2001)

2.5.7 – Fifteen Months and Beyond

First words tend to vary by child and by language. In many languages, object words (nouns) are more common, while in others (e.g. Korean), action words (verbs) are more

common. (MacWhinney 1998) For English, Pinker provides the following description of common first words:

About half the words are for objects: food (*juice, cookie*), body parts (*eye, nose*), clothing (*diaper, sock*), vehicles (*car, boat*), toys (*doll, block*), household items (*bottle, light*), animals (*dog, kitty*), and people (*dada, baby*) ... There are words for actions, motions, and routines like *up, off, open, peekaboo, eat, and go*, and modifiers, like *hot, allgone, more, dirty, and cold*. Finally, there are routines used in social interaction, like *yes, no, want, bye-bye, and hi*—a few of which, like *look at that* and *what is that*, are words in the sense of listemes (memorized chunks). (Pinker 2000, 270)

Some researchers believe that first words are acquired for the things that adults focus on in the child's presence, while others believe that first words are acquired for the things that are important to the child. (MacWhinney 1998)

The first words spoken by a child do not have symbolic meaning in the sense that adult language does. Rather, they are mere extensions to early nonlinguistic concepts. These extensions, along with emerging nonverbal gestures, allow the child to become an active participant in his/her social and cultural environment. (Norris and Hoffman 2002)

Somewhere around eighteen months of age, lexical acquisition explodes. Vocabulary growth occurs in bursts, but averages range from two words per hour to nine words per day. Children start to combine words into two word utterances and somewhere between eighteen and thirty-six months of age, children begin to speak using clauses and phrases. (Pinker 2000; Calvin and Bickerton 2001)

As the computational model introduced in chapter 4 only acquires a basic protolanguage, no attempt will be made to summarize the full emergence of language in children. Interested readers are referred to the works by Pinker, Nelson, Norris and Hoffman, and Calvin and Bickerton in the references section for more information.

2.6 – Summary

This chapter has provided an overview of several of the traditional views on early language and cognitive development in children. It has also summarized two modern integrated models of development, the Experiential Model and the SDS Model. In addition, a chronology of development was provided for the first year-and-a-half of life.

The next chapter provides an overview of several bottom-up computational models of perceptual grounding and human language acquisition.

CHAPTER 3 – COMPUTATIONAL MODELS OF PERCEPTUAL GROUNDING AND LANGUAGE ACQUISITION

3.1 – Introduction

While still a young field with few practitioners, some excellent work has been done to develop bottom-up computational models capable of grounded perception and lexical acquisition. This chapter will highlight some of the existing models, summarizing their techniques and major contributions.

3.2 – Cross-Situational Techniques for Lexical Acquisition

Throughout the 1990's, Siskind (Siskind 1992; 1997) has established algorithms to map words to symbolic representations of their meanings. For example, given the utterance, "*John walked to school.*" and a symbolic representation of the event, "GO(**John**, TO(**school**)),² his system would learn the mappings, "*John* → **John**, *walked* → GO(x, y), *to* → TO(x), and *school* → **school**." For a given utterance, the system proceeds as follows:

1. For each word in the utterance, some sense of that word is chosen from a known lexicon.
2. The symbolic conceptual expressions representing the sense chosen for each word are returned in an unordered list to a composition routine.
3. The composition routine builds a constrained list of possible symbolic utterance meanings from the individual symbolic word meanings.
4. The lexical acquisition portion of the system is presented with the original utterance and the set of possible symbolic utterance meanings.
5. With no knowledge of the source lexicon or the composition routine, the system begins to learn and build a target lexicon in two phases:

- a. Map each word to the set of conceptual *symbols* used to express the meaning of that word.
- b. Construct the appropriate conceptual *expressions* from the learned set of conceptual symbols.

Since each word in the source lexicon can contain multiple senses, the system must resolve lexical ambiguity. In addition, the system must resolve interpretive ambiguity because the composition routine returns multiple symbolic meanings for each utterance. Siskind's system can also handle the noisy situation where none of the symbolic meanings returned by the composition routine map to the current utterance.

To perform the word-to-meaning mappings, Siskind utilizes cross-situational learning. Basically, this means that the system resolves mappings only after being presented with multiple utterance/symbolic concept sets representing multiple *situations*. By drawing inferences about word mappings from multiple uses, the system is able to determine the correct symbolic mappings. Additional utterances aid in the resolution of expression mappings.

Siskind presents two versions of his cross-situational algorithm. The more complex version handles lexical ambiguity and noise, and is beyond the scope of this summary. (Siskind 1997) In the simpler version, the algorithm operates in two phases. Phase one establishes word-to-conceptual symbol mappings using a *necessary* and a *possible* conceptual symbol set and four inference rules. The sets for this phase will be referred to as $N(w)$ and $P(w)$ respectively. For each word in a new utterance, $N(w)$ is initialized to the empty set and $P(w)$ is initialized to the universal set of conceptual symbols. The following rules are then applied repeatedly to the two sets, adding symbols to $N(w)$ and removing them from $P(w)$ until the two sets converge.

Rule 1 *Ignore those utterance meanings that contain a conceptual symbol that is not a member of $P(w)$ for some word symbol w in the utterance. Also ignore*

those that are missing a conceptual symbol that is a member of $N(w)$ for some word symbol w in the utterance. (Siskind 1997, 57)

Rule 2 *For each word symbol w in the utterance, remove from $P(w)$ any conceptual symbols that do not appear in some remaining utterance meaning. (Siskind 1997, 58)*

Rule 3 *For each word symbol w in the utterance, add to $N(w)$ any conceptual symbols that appear in every remaining utterance meaning but that are missing from $P(w')$ for every other word symbol w' in the utterance. (Siskind 1997, 58)*

Rule 4 *For each word symbol w in the utterance, remove from $P(w)$ any conceptual symbols that appear only once in every remaining utterance meaning if they are in $N(w')$ for some other word symbol w' in the utterance. (Siskind 1997, 59)*

Once $P(w)$ and $N(w)$ converge, the algorithm begins the second phase, which determines the correct word-to-conceptual expression mappings using a *necessary* conceptual symbol set and a *possible* conceptual expression set. The sets for this phase will be referred to as $N(w)$ and $D(w)$ respectively. The following two additional rules are repeatedly applied until there is only one conceptual expression for each word in the original utterance.

Rule 5 *Let $RECONSTRUCT(m, N(w))$ be the set of all conceptual expression that unify with m , or with some subexpression of m , and that contain precisely the set $N(w)$ of non-variable conceptual symbols. For each word symbol w in the utterance that has converged on its actual conceptual-symbol set, remove from $D(w)$ any conceptual expressions not contained in $RECONSTRUCT(m, N(w))$, for some remaining utterance meaning m . (Siskind 1997, 60)*

Rule 6 *If all word symbols in the utterance have converged on their actual conceptual-symbol sets, for each word symbol w in the utterance, remove from $D(w)$ any conceptual expressions t , for which there do not exist possible conceptual expressions for the other word symbols in the utterance that can be given, as input, to $COMPOSE$ (the composition routine), along with t , to yield, as its output, one of the remaining utterance meanings. (Siskind 1997, 61)*

Note that some words such as “the” in an original utterance are determined to have no meaning and are ignored. Also the conceptual expression rules can sometimes take an

excessively long time to run, in which case, their execution is terminated after some pre-determined time limit.

Siskind's system successfully bootstraps a lexicon to 95% convergence in the presence of noise. His system scales well using vocabularies of 1,000 to 10,000 words. Siskind's system does have some limitations. It cannot learn idiomatic or metaphoric meaning directly, but can learn, in the presence of such meanings, by treating them as noise. In addition, polysemy, which is a word with multiple, related senses, has been omitted from the model. Finally, the inference rules developed by Siskind restrict the form of compositional semantics to argument substitution. (Siskind 1992; 1997)

3.3 – Force Dynamics and Event Logic for Grounded Event Recognition

In distinct but related research, Siskind (Siskind 1992; 2000; 2001; Siskind and Morris 1996) has developed several software systems to classify and describe dynamic events. In 1992, he described ABIGAIL, a system that constructs semantic descriptions of events occurring in computer-generated stick-figure animations. ABIGAIL *perceives* events by detecting support, contact, and attachment using counterfactual simulation. (Siskind 1992)

Using a subsequent system named HOWARD, Siskind and Morris built event representations based on real video. HOWARD produces hidden Markov models (HMMs) of the motion profiles of the objects involved in an event. (Siskind and Morris 1996)

Siskind's most recent approach has been to use event-logic to describe changes in support, contact, and attachment, which he now terms *force-dynamics*. In comparing motion profiles and force-dynamics, Siskind states,

Event recognition is a process of classifying time-series data. In the case of motion profile, this time-series data takes the form of relative-and-absolute positions, velocities, and accelerations of the participant objects as a function of time. In the case of force dynamics, this time-series data takes the form of the

truth values of force-dynamic relations between the participant objects as a function of time. (Siskind 2001, 33)

Siskind's latest implementation is called LEONARD. It uses a camera to capture a sequence of images and then processes that sequence using three subroutines:

1. Segmentation-and-Tracking – places a polygon around the objects in each frame
2. Model-Reconstruction – builds a force dynamic model of each polygon scene, determining grounding, attachment, and depth/layering
3. Event-Classification – determines over which intervals various primitive event types are true and from that data, over which intervals various compound event types are true

Advantages of using event logic for force dynamics include less sensitivity to variance in event occurrences, correct classification of events in the presence of unrelated objects, processing of hierarchical events through temporal and spatial segmentation, and finally, the detection of non-events. (Siskind 2000; 2001)

3.4 – X-Schemas, F-Structs, and Model-Merging for Verb Learning

Bailey (Bailey 1997; Bailey, et. al. 1997; 1998) has developed a computational model of the role of motor control in verb acquisition. He argues that proprioception, which is knowledge of the body's own state, is linked to the acquisition of action verbs. In fact, he maintains that grounding action verbs in the motor-control system constrains the variety of lexical action categories and makes verb acquisition tractable. Bailey introduces the executing schema (x-schema) as a mechanism that can represent and carry out verbal commands, and feature structures (f-structs) as a mechanism for linking x-schema activities to related linguistic features.

X-schemas are formal representations of sequences of motor control actions. In Bailey's model, x-schemas are modeled as Petri nets with extensions to handle the passing of parameters.

Petri nets provide a mechanism for representing sequential and/or temporal sequences of actions. They can be described in terms of *places*, *transitions*, *tokens*, and *directed connections*, where *places* contain *tokens*, and *directed connections* link *places* and *transitions*. At any time, the *state* of a Petri net is represented by the *places* that contain *tokens*. Bailey's notation for Petri nets and thus x-schemas uses circles to represent *places*, rectangles to represent *transitions*, and dots to represent *tokens*. He uses labeled *transitions* to denote *action*, and *transitions* with a double vertical bar, ||, to denote *concurrency*.

Figure 10 is taken from Bailey (Bailey 1997) and displays his SLIDE x-schema, which describes how an arm and hand would slide some object across a table. Starting from the left, the arm approaches the object to be moved as the hand concurrently shapes itself based on the size of the object. Once the hand is in contact with the object, the arm begins to move it horizontally toward the goal position. While sliding, the hand will tighten its grip if slippage is detected. Horizontal movement continues until the goal position is reached.

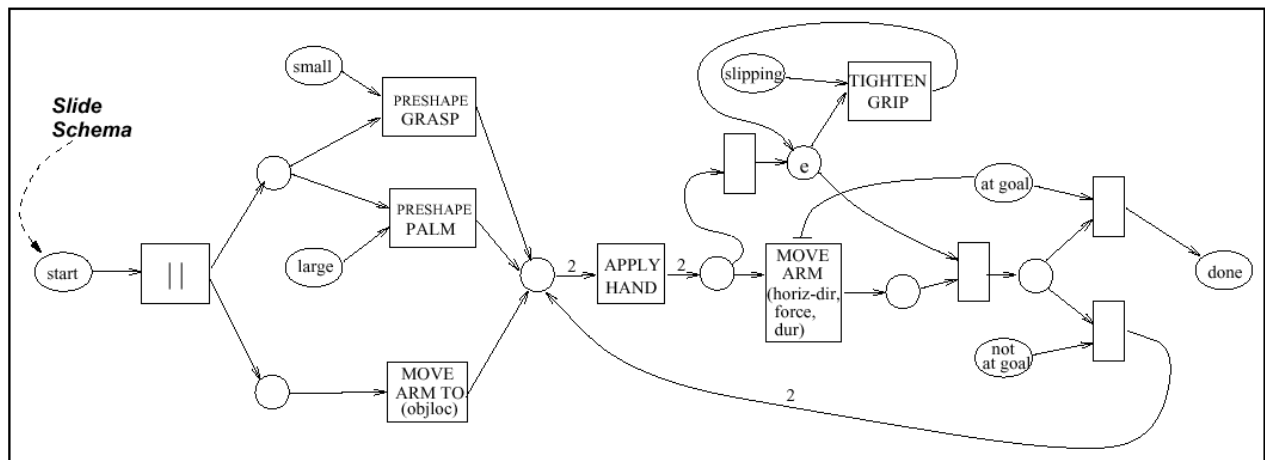


Figure 10. SLIDE X-Schema (Bailey 1997, 33)

In order to connect x-schemas to verbs, the linking feature structure (f-struct) is introduced. The f-struct is an intermediate set of features that allows a layer of abstraction between the individual motions of an action and the action verb that describes them. An f-struct

is a list of feature-value pairs represented in a table with two rows. Each pair maps to a column with the feature located in the top row and the value in the bottom row. Bailey experientially determined a list of twelve features for his system comprised of eight motor control features and four perceived world state features.

Bailey's system performs verb acquisition using an algorithm that develops a lexicon of word senses based on a training set of verbs and linking f-structs summarizing that verb. "The basic idea is to start with a lot of very specific senses for each verb, and then gradually merge them together to form a smaller set of more general senses." (Bailey 1997, 93-94) Verb learning becomes an optimization problem to find the best possible lexicon given the training examples. Bailey terms this approach for merging word senses, *model-merging*, and implements a solution using a hill-climbing algorithm.

As a final note, Bailey's computational model was implemented in a system called VerbLearn. It was designed to operate in a virtual environment via an interface with *Jack*, a commercial human simulation system. The environment for the model was limited to the actions of a single arm and hand manipulating simple geometric objects on a table. (Bailey 1997; Bailey, et. al. 1997; 1998)

3.5 – Cross-Modal Acquisition of Shape and Color Words

The CELL system developed by Roy (Roy 1999; 2000a; 2000b) is a grounded, robot-based system that has been shown to learn shape and color words based on multi-modal perceptual information. CELL was simultaneously presented with a series of objects and an audio description for each. The dataset was collected from audio recordings of speech during play sessions between caregivers and seven to eleven month-old infants. During these sessions,

each caregiver and infant played with toys from seven categories including balls, shoes, keys, cars, trucks, dogs, and horses.

CELL took pictures of each toy in the dataset from multiple angles and abstracted a *visual model* from histogram representations of shape and color. Next, it abstracted a *speech model* based on phonemic estimation for the utterances from the play sessions with each toy. Based on these models CELL attempted to correlate the reoccurring features in the speech models with reoccurring features in the visual models for shape and color.

The correlation process employed by CELL uses a combination of short-term memory (STM) and long-term memory (LTM) mechanisms. As the visual and speech models are created for each object, they are loaded into the STM.

The STM has a capacity of five utterances, corresponding to approximately 20 words of infant-directed speech. As input is fed into the model, each new [utterance,shape,color] entry replaces the oldest entry in the STM. A short-term *recurrence filter* searches the contents of the STM for recurrent speech segments which occur in matching visual contexts. (Roy 2000b, 9-10)

Recurring speech segments are paired with the visual models and moved to the LTM. In the LTM, the speech segments and visual models are merged into prototypical categories representing word-meaning mappings.

In addition to the *acquisition* mode described above, CELL has an *understand* mode and a *generate* mode. In the *understand* mode, it attempts to locate an object in its environment based on a spoken description. In the *describe* mode, it attempts to generate a spoken description for a novel object.

It has been shown that when CELL is able to correctly segment complete English words from the audio signal, it can correlate them with the proper visual models 57±10% of the time. (Roy 1999; 2000a; 2000b)

3.6 – Acquisition of Words and Grammar for Spatial Description Tasks

In related research, Roy (Roy 2000b) has developed a system called DESCRIBER, which acquires the words and grammar necessary to produce spatial descriptions of computer generated images. The images produced by the computer each contain ten non-overlapping rectangles that randomly vary in size, position, and color. DESCRIBER acquires its language based on a training set of transcribed human descriptions of a target rectangle in each image.

To evaluate DESCRIBER, the description roles are reversed. DESCRIBER generates descriptions of target rectangles, and human test subjects attempt to identify those targets. When tested, human subjects were able to identify the correct target 81.3% from DESCRIBER's descriptions and 89.8% of the time from other human descriptions.

Roy is in the process of extending DESCRIBER to incorporate a camera-based computer vision system. He has also mentioned adding a speech recognition system to make the acquisition process more realistic. (Roy 2000b)

3.7 – Social Learning of Language and Meaning

Steels and Kaplan (Steels and Kaplan 2000) have integrated a language acquisition model into customized software for Sony AIBO™ robotic dogs. The model can acquire simple object words based on social mediation.

The AIBO robot uses integrated computer vision, speech recognition and speech synthesis systems to perceive its environment and to interact with a human mediator. It is preprogrammed to recognize and respond to several action commands including “stand up,” “look,” and “what is it?,” and several feedback commands including “yes,” “no,” and “good.”

The AIBO's ability to learn names for three colored objects (a red ball, a yellow puppet, and a small AIBO imitation) was evaluated in three modes: *strong interaction*, *observational*

learning with supervision, and *unsupervised learning*. In the first mode, the human moderator used physical gestures and the AIBO's command set to draw its attention to the three objects. The moderator then supplied names for the objects. The AIBO generated color histogram representations of the objects in its perceptual field and attempted to resolve them to the supplied names. The second mode was identical to the first, except that the moderator did not use any sort of physical feedback. In the third mode, the AIBO was completely unsupervised and had to interact with the three objects on its own.

After training in the *strong interaction* mode, the AIBO was able to correctly classify the three objects with an 82% success rate. This compares with a 59% success rate for the *observational learning with supervision* mode. As names were not available for the *unsupervised learning* mode, the AIBO's ability to cluster objects was used to measure performance instead. It was only able to correctly cluster 30% of the objects. These results make a fairly strong case for the role of social mediation in learning. (Steels and Kaplan 2000)

3.8 – Summary

This chapter has summarized several bottom-up models for grounded perception and language acquisition. Table 1 provides a summary of the features in each model. Siskind's model for cross-situational word learning handles multiple parts of speech, but maps words to symbolic representations of rather than actual perceptual data. (Siskind 1992; 1997) His research on event recognition incorporates real videos, but deals only with event naming and does not attempt to incorporate full object recognition. (Siskind 1999; 2000; 2001) In a similar fashion, Bailey's VerbLearn software focuses primarily on verbs, using proprioception to perceive events in a virtual environment. (Bailey 1997) In contrast, the research by Steels and Kaplan involving Sony AIBO™ robots focuses on object naming based on histogram

representations of the AIBO's vision system, and it relies heavily on social mediation and feedback. (Steels and Kaplan 2000) Roy's research with the CELL system investigates various approaches to speech segmentation and utilizes a computer vision system to aid in the lexical segmentation of color and shape words from an audio signal. In addition, CELL can attempt to find an object in its environment corresponding to a spoken description or generate a spoken description of a specified object. In related work, Roy's DESCRIBER system acquires and generates noun phrases describing computer-generated scenes of colored, non-overlapping rectangles. While DESCRIBER can generate spatial descriptions, neither of Roy's systems attempts to incorporate dynamic relationships or verb learning. (Roy 1999; 2000a; 2000b)

Table 1. Comparison of Existing Computational Models

Model / Feature	Bottom-Up	Parts-of-Speech	Syntax	Perception	Speech Processing	Socially Mediated	Source Available
Bailey (VerbLearn)	requires explicit training	action verbs	no	virtual proprioception	text	no	yes
Roy (CELL)	yes	shape and color words	no	vision (histogram-based)	audio	no	no
Roy (DESCRIBER)	requires explicit training	spatial description words	yes	virtual vision	text in / audio out	no	no
Siskind (HOWARD & LEONARD)	requires explicit training	event labels	no	vision (segmentation-based)	text	no	yes
Siskind (word-to-meaning mappings)	yes	multiple	no	virtual "meaning" symbols	text	no	yes
Steels & Kaplan (AIBO)	utilizes base lexicon	object nouns	no	vision (histogram-based)	audio	yes	no

Based upon the developmental foundations presented in chapter 2 and the computational foundations presented in chapter 3, the next chapter provides a general overview of Experience-Based Language Acquisition (EBLA), a new computational model of early language acquisition. A technical discussion detailing the implementation of EBLA follows in chapter 5.

CHAPTER 4 – OVERVIEW OF THE EXPERIENCE-BASED LANGUAGE ACQUISITION MODEL

4.1 – Introduction

This chapter introduces the Experience-Based Language Acquisition (EBLA) Model, a new experience-based computational model of early language acquisition that operates in a bottom-up fashion. The fundamental goal for the EBLA Model is to acquire a childlike protolanguage grounded in perceptual experience. EBLA is an experience-based model in that it detects and abstracts basic information about objects and their relationships using a computer vision system. It is a language acquisition model in that it attempts to resolve protolanguage descriptions to internal representations of its experiences. Finally, it is a bottom-up model in that it is not preprogrammed with any information about objects, object-object relations, or lexical mappings.

A secondary goal for the EBLA Model is to establish a framework for future research and experimentation. EBLA is cross-platform, well-documented, and it has been developed using freely available technologies whenever possible. It has been designed in a modular fashion so that it can be understood and extended by other researchers.

The EBLA Model operates by observing a series of “experiences” in the form of short movies. Each movie contains a single event such as an arm/hand picking up a ball, and takes the form of either an animation or an actual video. The model detects any significant objects in each movie and determines what, if any, relationships exist among those objects. This information is then stored so that repeatedly occurring objects and relations can be identified across multiple experiences.

As part of each experience, EBLA receives a textual description of the event taking place. These descriptions are comprised of protolanguage such as “hand pickup ball.” To acquire this

protolanguage, EBLA must correlate the lexical items in the descriptions to the objects and relations in each movie. Figure 11 provides a graphical representation of the method used by EBLA to process experiences.

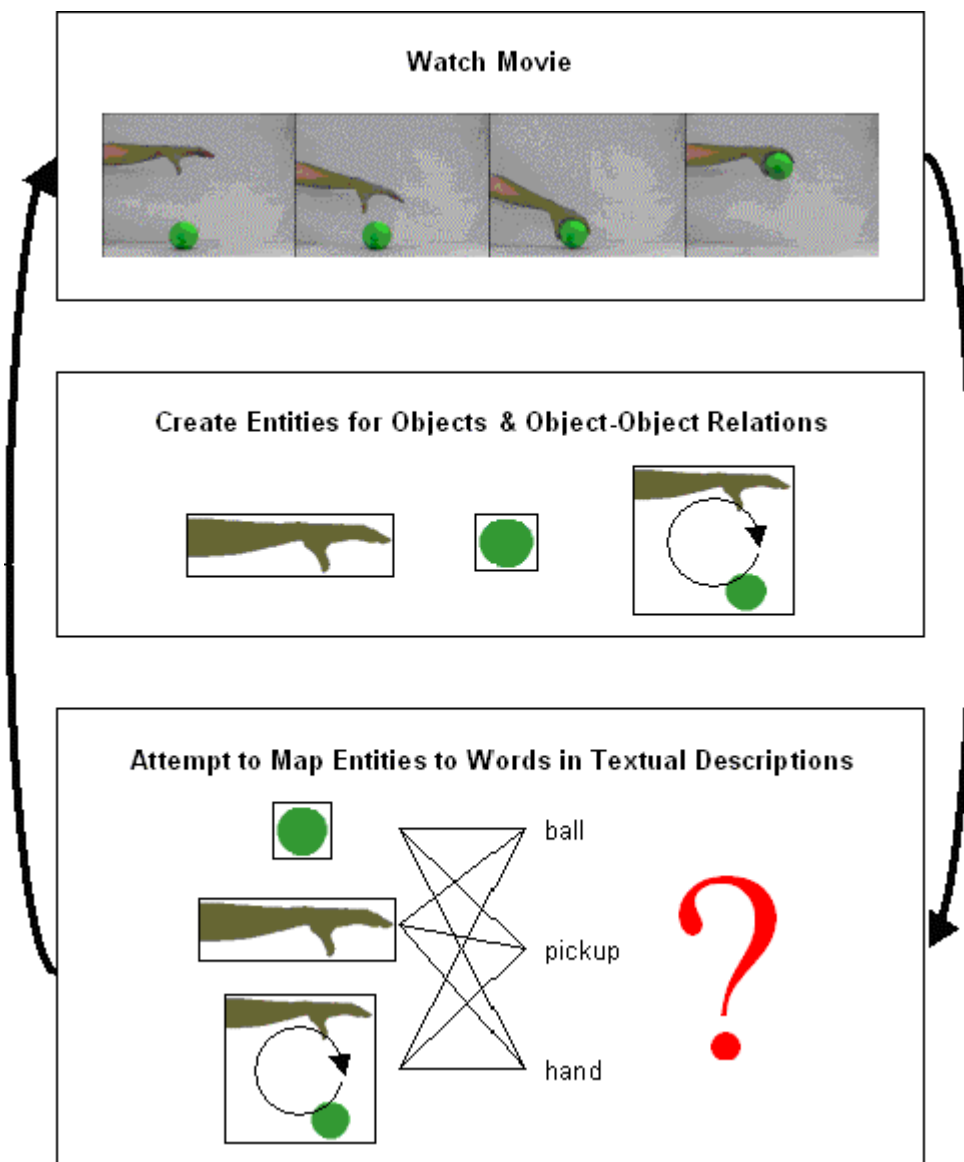


Figure 11. Method Used by EBLA to Process Experiences

4.2 – Developmental Basis for Model

The computational approaches used in EBLA are based in part on what is known about cognitive development and language acquisition in children. In particular, Nelson’s Experiential

Model has heavily influenced the current research. It should be noted, however, that only the earliest stages of the Nelson's model have been incorporated into the current work. The movies and descriptions processed by EBLA are simplistic in comparison to the real-life events experienced by a child. Furthermore, there has been no attempt to incorporate social or cultural mediation into EBLA.

Within the EBLA Model, each movie/description combination presented to the model is considered to be an *experience*. EBLA attempts to build an *event representation* for each experience, and then attempts to incorporate language into that representation. Initially the individual lexical items in each description have no meaning on their own and are associated with the entire experience. Over time, the model is able to correlate words with particular objects or relations. As the model is exposed to more instances of a word, it integrates variations in experiences and begins to decontextualize both objects/relations and lexical data.

Since the EBLA Model deals only with protolanguage, it avoids the dispute between the empiricist and nativist camps about the extent to which the capacity for language is genetically encoded. It seems that there is little debate that the newborn has little or no knowledge of his/her environment prior to birth, but has some genetic capacity to perceive his/her environment and to organize that perceptual information. In a similar fashion, the model has been given only basic perceptual and organizational capacities. Information regarding particular objects and relations is not preprogrammed.

The current research only investigates how the very first words might be incorporated into perceptual knowledge. The nature/nurture conflict enters the picture as the child moves from protolanguage to language and begins to make use of syntax. Although any determination

of how a child's full mastery of language emerges is well beyond the scope of this work, the basic framework provided by EBLA may conceivably facilitate such research in the future.

4.3 – Computational Basis for Model

EBLA builds on several computational concepts established by other researchers. Both Siskind and Bailey have created systems to analyze and label events comprised of actions on simple objects. Although a different algorithm is used, both Siskind's LEONARD system and EBLA use image segmentation as the basis for their vision systems. The attribute-value system used to represent entities in EBLA is based on Bailey's linking feature structure (f-struct) notation. (Bailey 1997; Siskind 2001) In addition, the cross-situational learning algorithms employed by EBLA to map lexical items to objects and object-object relations are based directly on the inference and exclusivity techniques used by Siskind to map words to symbolic meanings. (Siskind 1997) EBLA differs from similar computational models in that it is one of the first (if not *the* first) to integrate noun *and* verb acquisition using a grounded perceptual system.

4.4 – Model Abstractions and Constraints

In order to implement the EBLA Model in a reasonable amount of time, it has been constrained in several ways. First, the model's perceptual capabilities are limited to a two-dimensional vision system that reduces objects to single color polygons. There is no attempt to incorporate auditory, olfactory, tactile-proprioceptive, or gustatory perceptual capabilities. While this greatly simplifies the model, it likely hinders it, since all of the senses have an impact on the human conceptual system. Future incorporation of additional senses should only enhance the model's internal representations and its capacity to acquire language.

Second, the model has not been provided with any audio processing capabilities. Because of this, all experience descriptions presented to or generated by EBLA are textual.

Without this abstraction, the model would require a speech recognition system to separate the auditory signal into phones and then transform those phones into text. Since speech-processing technology is imperfect, providing audio descriptions of each experience could inadvertently introduce noise. Moreover, incorporation of a speech module would add an unnecessary layer of complexity. Such a module could easily be added at a later time as technology improves (see Roy 1999 for additional information on speech segmentation).

Third, the model only attempts to acquire a protolanguage of nouns and verbs. Thus, syntax, word order, punctuation, etc. do not apply. This conforms with early human language acquisition since children do not begin to use phrases and clauses until somewhere between eighteen and thirty-six months of age. (Calvin and Bickerton 2001) This may ultimately limit the extent to which the model acquires language since some believe that syntactic frames facilitate vocabulary growth. (MacWhinney 1998) For the simple experiences presented to EBLA, however, lack of syntactic structure does not present a problem.

The final constraint on EBLA is that it only operates in an unsupervised mode. This means that the model does not receive any sort of feedback regarding its accuracy. This is definitely a worst-case scenario since children receive frequent social mediation in all aspects of development. While such mediation likely aids children and accelerates their acquisition process, success of the current model may indicate that such feedback plays more of a motivational role.

4.5 – Experiences Processed by the EBLA Model

The experiences processed by the EBLA Model are based on the simple spatial-motion events used by Siskind and Bailey, and take the form of either animations or real videos. Experiences are delivered to EBLA as digital video files. Each experience contains an arm/hand

performing some simple action on a variety of objects. For the animations, the actions include *pickup*, *putdown*, *touch*, and *slide*, and the objects include a green ball and a red cube (see figure 12). For the real videos, the actions include *push*, *pull*, *slide*, *touch*, *tipover*, *roll*, *pickup*, *putdown*, *drop*, and *tilt*, and the objects include several colored bowls, rings, and cups, a green ball, a dark blue box, a blue glass vase, a red book, and an orange stuffed Garfield cat (see figure 13).

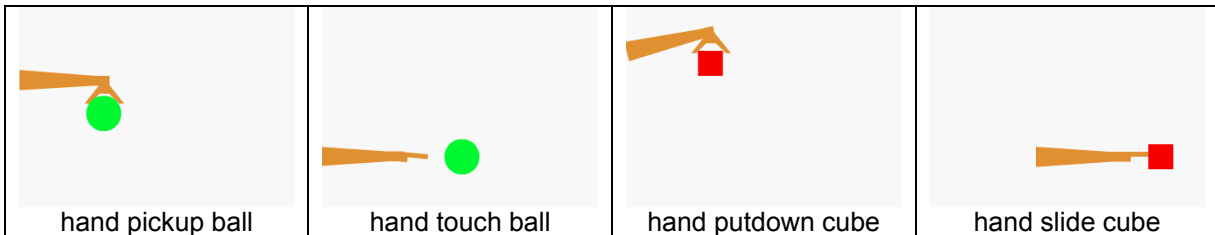


Figure 12. Frames from Various Animations Processed by EBLA

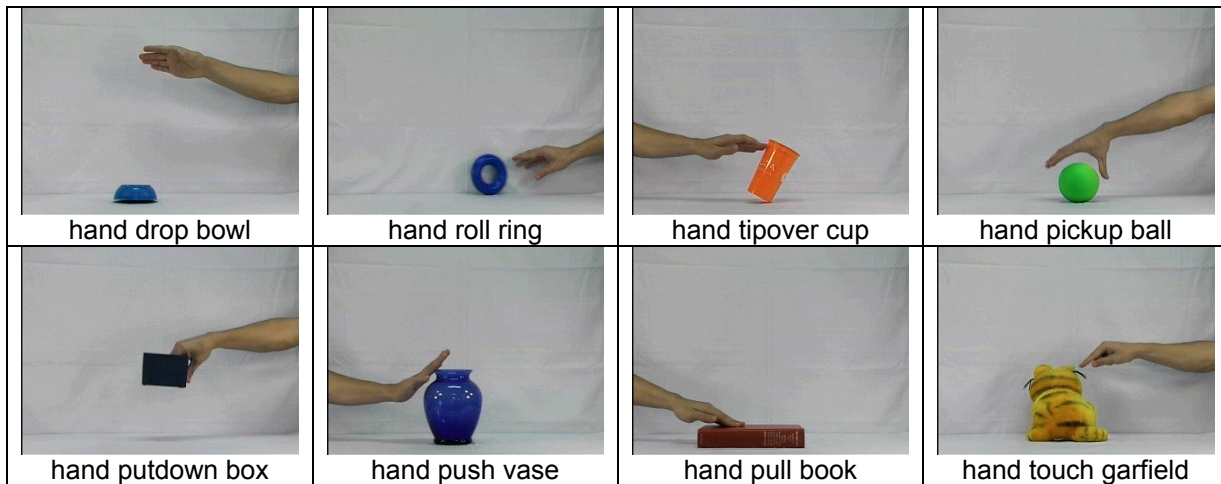


Figure 13. Frames from Various Videos Processed by EBLA

A more technical discussion of how the experiences for EBLA were generated is provided in chapter 6.

4.6 – Entity Recognition

The EBLA Model has a basic perceptual system, which allows it to “see” the significant objects in each of its experiences. EBLA calculates a set of static attribute values for each object

and a set of dynamic attribute values for each object-object relation. The sets of attribute-value pairings are very similar to the linking feature structures (f-structs) used by Bailey. (Bailey 1997) Each unique set of attribute values defines an *entity*, and is compared to the entities from prior experiences. If there is a match within a certain variance, the current entity is merged with the existing entity, creating a more prototypical entity definition. Otherwise, a new entity definition is established. EBLA's entities roughly correspond to the child's *internal representations* of perceptual information, and a set of entities for a given experience roughly corresponds to an *event representation*.

Both the object and relation attributes for EBLA were determined experimentally based on data available from the computer vision system. There are five object attributes and seven relation attributes. A list of the attributes along with basic definitions is provided in table 2. A more technical discussion of the entity-attribute system can be found in section 5.8.

Table 2. Entity Attributes Calculated by EBLA

Entity	Type	Description
area	object	area (in pixels) of a given object
grayscale value	object	grayscale color of object (0-255)
number of edges	object	number of edges on polygon tracing object
relative centroid (x)	object	horizontal coordinate of object's center of gravity relative to the width of a bounding rectangle around the object
relative centroid (y)	object	vertical coordinate of object's center of gravity relative to the height of a bounding rectangle around the object
contact	relation	Boolean value indicating if two objects are in contact with one another
x-relation	relation	indicates whether one object is to the left of, on top of, or to the right of another object
y-relation	relation	indicates whether one object is above, on top of, or below another object
delta-x	relation	indicates whether the horizontal distance between two objects is increasing, decreasing, or unchanged
delta-y	relation	indicates whether the vertical distance between two objects is increasing, decreasing, or unchanged
x-travel	relation	indicates direction of horizontal travel for both objects
y-travel	relation	indicates direction of vertical travel for both objects

Currently, object entities are defined using all of the object attributes, and relation entities are defined using all of the relation attributes. There is no mechanism to drop attributes that may not be relevant to a particular entity. For example, grayscale color value may not have anything to do with whether or not an object is a ball, but EBLA would likely create separate entities for a light-colored ball and a dark-colored ball. A variation of the model-merging algorithm employed by Bailey could be applied to drop attributes unrelated to the *essence* of a particular entity. Because EBLA currently uses a limited number of attributes, dropping any would likely lead to overgeneralization of entities, but with more attributes, it could be a very useful mechanism. This enhancement is discussed further in chapter 7.

Entities in EBLA are atomic in nature in that there are no compound entities. Object entities are based on individual objects and relation entities are based on single object-object pairings. For example, because EBLA's vision system is based on color image segmentation, it would most likely create separate entities for a person's head, arms, legs, and torso despite the fact that these entities are always in contact and act as a unified whole. In a similar fashion, EBLA perceives the stacking of several objects on top of one another as a series of distinct relation entities. EBLA could conceivably be extended to recognize compound entities by linking object entities that appear to be permanently connected across an entire experience (e.g. a head and torso) and linking relation entities that involve common object entities (e.g. stack or play).

4.7 – Lexical Acquisition

Once EBLA has generated entities for the objects and object-object relations in each experience, its final task is to map those entities to the lexemes (words) in protolanguage descriptions of each experience. Protolanguage was chosen because it is the first type of

language acquired by children. The particular variety of protolanguage used for the EBLA's experience descriptions has the following characteristics:

1. Word order is not important, although the descriptions provided to EBLA are generally in the format: *subject-manipulation-object* (e.g. "hand touch ball").
2. Verbs paired with particles are combined into a single word (e.g. "pick up" becomes "pickup").
3. Words are not case-sensitive (although there is an option in EBLA to change this).
4. Articles (e.g. "a," "an," "the") can be added to descriptions, but are generally uninterpretable by EBLA.

It should be noted that EBLA is not explicitly coded to ignore articles, but since they are referentially ambiguous when considered as individual, unordered lexemes, EBLA is unable to map them to entities. Adding articles to the protolanguage descriptions generally slows down EBLA's average acquisition speed.

In order to map the individual lexemes in the protolanguage descriptions to the entities in each experience, EBLA must overcome referential ambiguity. This is because EBLA operates in a bottom-up fashion and is not primed with any information about specific entities or lexemes. If the first experience encountered by EBLA is a hand sliding a box with the description "hand slide box," it has no idea whether the lexeme "hand" refers to the *hand* object entity, the *box* object entity, or the *slide* relation entity. This same referential ambiguity exists for the "slide" and "box" lexemes. EBLA can only overcome this ambiguity by comparing and contrasting the current experience with future experiences. This process of resolving entity-lexeme mappings is a variation of the cross-situational learning employed by Siskind. (Siskind 1992; 1997)

For each experience, two lists are created to hold all of the unresolved entities and lexemes. EBLA attempts to establish the correct mappings for these lists in three stages:

1. Lookup any known resolutions from prior experiences.
2. Resolve any single remaining entity-lexeme pairings.
3. Apply cross-situational learning, comparing unresolved entities and lexemes across all prior experiences, repeating stage two after each new resolution.

To perform the first stage of lexical resolution, EBLA reviews known entity-lexeme mappings from prior experiences. If any match both an entity and lexeme in the current experience, those pairings are removed from the unresolved entity and lexeme lists.

The second stage operates on a simple process of elimination principal. If at any point during the resolution process both the unresolved entity and lexeme lists contain only a single entry, it is assumed that those entries map to one another. In addition, prior experiences are searched for the same entity-lexeme pairing and resolved if found. Since resolving mappings in prior experiences can generate additional instances of single unmapped pairings, the entire second stage is repeated until no new resolutions are made.

The third and final stage of resolution is by far the most complex and involves a type of cross-situational inference. Basically, by comparing the unresolved entities and lexemes across all experiences in a pair wise fashion, EBLA can infer new mappings. If the cardinality of the intersection or difference between the unmapped entities and lexemes for a pair of experiences is one, then that intersection or difference defines a mapping. In more formal terms:

1. Let i and j be any two experiences, $i \neq j$.
2. Let E_i and $E_j \in$ unmapped entities for i and j respectively.
3. Let L_i and $L_j \in$ unmapped lexemes for i and j respectively.

4. If $|\{E_i \cap E_j\}| = 1$ and $|\{L_i \cap L_j\}| = 1$ then $\{E_i \cap E_j\}$ maps to $\{L_i \cap L_j\}$.
5. If $|\{E_i \setminus E_j\}| = 1$ and $|\{L_i \setminus L_j\}| = 1$ then $\{E_i \setminus E_j\}$ maps to $\{L_i \setminus L_j\}$.
6. If $|\{E_j \setminus E_i\}| = 1$ and $|\{L_j \setminus L_i\}| = 1$ then $\{E_j \setminus E_i\}$ maps to $\{L_j \setminus L_i\}$.

To demonstrate how all three stages work together, consider the following example. If the model was exposed to an experience of a hand picking up a ball with the description “hand pickup ball” followed by an experience of a hand picking up a box with the description “hand pickup box,” it could take the set differences discussed in stage three for the two experiences to resolve the “ball” lexeme to the ball entity and the “box” lexeme to the box entity. Assuming that these were the only two experiences presented to the model, it would not be able to resolve “hand” or “pickup” to the corresponding entities because of referential ambiguity. If the model was then exposed to a third experience of a hand putting down a ball with the description “hand putdown ball,” it could resolve all of the remaining mappings for all three experiences. Using the technique discussed in stage one, it could resolve “ball” based on known mappings from the prior experiences. It could then take the set intersection with the unmapped items in either of the first two experiences to resolve “hand.” This would leave a single unmapped pairing in each of the three experiences, which could be resolved using the process of elimination discussed in stage two. Note that taking the set difference rather than the intersection between the third and first or second experiences would have worked equally well to resolve “hand pickup” and “hand putdown.”

4.8 – Summary

This chapter has presented an overview of the EBLA Model, discussing its links to existing developmental and computational works, the abstractions used in the model, the

experiences to be processed, and the techniques employed to perceive experiences and acquire language. The next chapter details the actual software implementation of EBLA.

CHAPTER 5 – IMPLEMENTATION OF THE EBLA MODEL

5.1 – Introduction

From the outset, there was no strong indication that implementing a system capable of correlating visually perceived objects and object-object relations to lexical items was even feasible. Success was very much dependent on obtaining or developing a computer vision system. The perceptual details available for analysis and storage were essentially unknown, so many details of the model including the methodology for storing objects and relations had to be determined on the fly.

Thus, for this work, the model and its implementation are very much one and the same. This chapter details that implementation. First, an overview of the programming language and database technologies involved in the EBLA software system is presented. This is followed by a detailed discussion of each of the object-oriented modules comprising EBLA.

5.2 – Technologies Involved

An underlying goal of this research has been to produce a software framework that can be easily understood by students interested in natural language processing or computational linguistics, and easily extended by students or researchers. To that end, EBLA has been developed using the Java programming language. The Java Software Development Kit (SDK) is freely available and platform-independent, and provides application program interfaces (API's) for video and image processing as well as database integration. In addition, EBLA has been documented using the JavaDoc documentation standards. JavaDoc allows thorough HTML documentation to be generated automatically from source code. A sample of the JavaDoc documentation is provided in appendix C.

EBLA has been designed to make extensive use of a PostgreSQL database system called **ebla_data**. PostgreSQL is an open-source, high performance, relational database with many enterprise-level features. Using a database provides a clean way to organize and analyze all of the information needed for the EBLA system. This includes run-time parameters, video paths and filenames, intermediate results, objects, object-object relations, lexical data, and lexical mappings. Figure 14 provides an abstracted view of EBLA’s general architecture and figure 15 provides a detailed diagram of the table and relation structure for the **ebla_data** database. The complete SQL used to construct the database is provided in appendix D. As the various components of EBLA are presented, the database will be discussed in further detail.

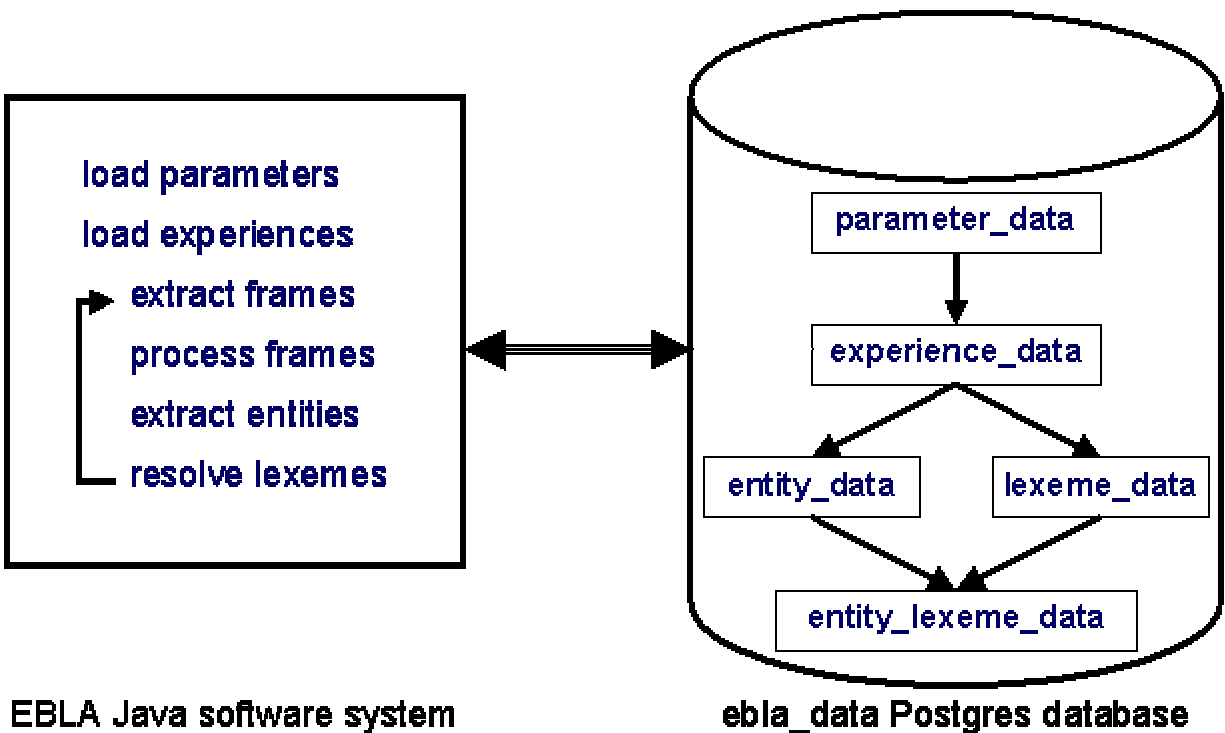


Figure 14. General Architecture of EBLA Software System

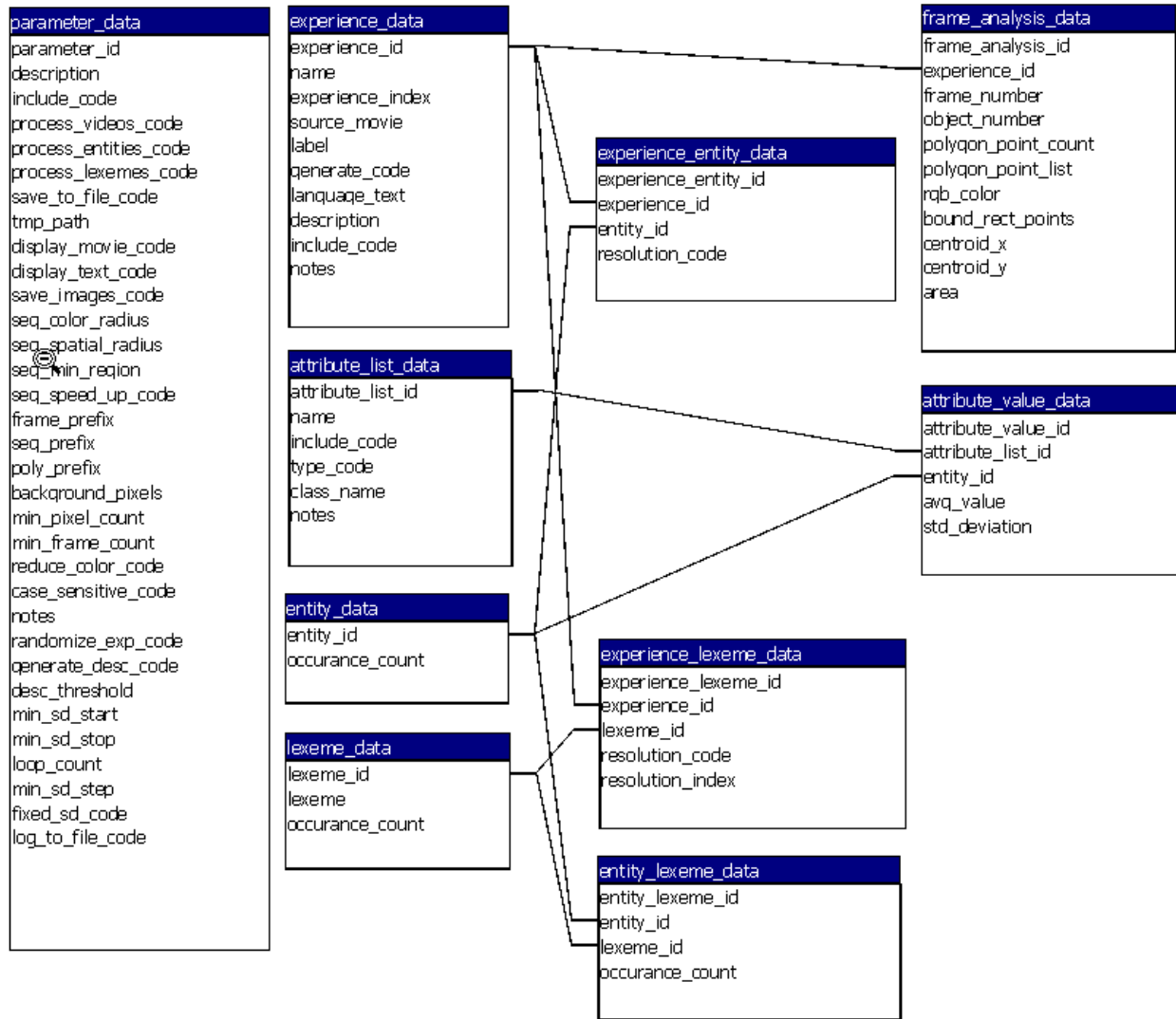


Figure 15. EBLA Database Tables and Relationships

5.3 – EBLA Executable Class

The majority of the EBLA software system is contained in the Java package *com.greatmindsworking.EBLA*. Java packages provide a way to use file directories to aid in code organization and minimize the risk of encountering naming conflicts with other software. Within the *com.greatmindsworking.EBLA* package, the system is run from an executable class called *EBLA*. *EBLA* contains a *main()* method so that it can be executed from a command

prompt. It takes a single, optional parameter containing a record identifier for the set of runtime parameters to retrieve from the **parameter_data** table in **ebla_data**.

The *main()* method parses the parameter, calls the appropriate *EBLA* constructor, and then calls the *processExperiences()* method. The constructor is responsible for establishing a connection with the database using the *DBConnector* class, retrieving the specified or default parameters using the *Params* class, and directing the intermediate results to either the display or a log file. The *processExperiences()* method is responsible for retrieving the list of experiences to process from the **experience_data** table and for instantiating all of the other top-level classes needed to process each experience. These top-level classes are briefly described in table 3 and detailed descriptions follow below.

Table 3. Top-Level Classes Instantiated by EBLA

Java Class	Description
DBConnector	establishes a connection to the EBLA database
Params	sets default runtime parameters or retrieves custom parameters from the database
FrameGrabber	extracts the individual frames from each experience (movie) as graphics files
FrameProcessor	segments each frame and stores intermediate frame data in the database
EntityExtractor	processes intermediate data and recognizes significant objects and object-object relations
LexemeResolver	parses the protolanguage associated with each experience and attempts to resolve it to the corresponding objects/relations

5.4 – Database Connection

The *DBConnector* class manages database connections for EBLA using the Java Database Connectivity (JDBC) API. *DBConnector* contains all of the information needed to establish a connection to the PostgreSQL database server including the IP address of the database server, the database name, and the username and password. Its constructor takes a single Boolean parameter that determines whether database queries are immediately committed to the database or cached and committed as a single transaction. *DBConnector* has two public methods: *getStatement()* and *commitChanges()*.

The *getStatement()* method is called throughout the EBLA system to retrieve the JDBC *Statement* objects needed to execute SQL queries. Since *Statement* objects cannot execute more than one query at a time, multiple instances are required to process nested queries. The *commitChanges()* method only needs to be called if queries against the database are being cached. It should be called whenever the cached queries are to be permanently committed.

As a final note, *DBConnector* contains a *main()* method that can be used for standalone testing to verify that the database connection is successful.

5.5 – Parameters

The second class called by the *EBLA* constructor is *Params*. *Params* sets defaults for all of the EBLA runtime options. If the user specifies a record identifier for the **parameter_data** table in the **ebla_data** database when launching EBLA, *Params* will overwrite the default values with the corresponding values in the database. This allows users to control EBLA without manipulating the source code. The runtime options determine which experiences will be processed, general object properties, where and how to generate intermediate results, and a number of adjustable settings for the computer vision system. A listing of all the parameters in the *Params* class along with a description of each is provided in table 4.

One parameter that deserves a special mention is *includeCode*. This parameter is matched against the **include_code** field in the **experience_data** table and is used to determine which experiences are processed by EBLA. Large blocks of experiences can be processed in parallel by dividing the records in the **experience_data** table into blocks with separate **include_code**'s. These blocks can then be processed on separate workstations by creating a **parameter_data** record for each block and then invoking a different set of parameters on each machine. This technique was successfully employed for processing some of the larger data sets.

Table 4. EBLA Parameters

Parameter	Default	Description
includeCode	1	code used to determine which movies from experience_data table to process
processVideos	true	Boolean flag indicating whether to process videos for current run
processEntities	true	Boolean flag indicating whether to process entities for current run
processLexemes	true	Boolean flag indicating whether to process lexemes for current run
logToFile	true	Boolean flag indicating whether to write results to screen or log file
randomizeExp	true	Boolean flag indicating whether to randomize the experiences processed by EBLA
generateDesc	false	Boolean flag indicating whether to generate descriptions for some experiences
descThreshold	7	number of experiences to process before trying to generate descriptions
minStdDevStart	5	starting minimum standard deviation for matching entities
minStdDevStop	5	stopping minimum standard deviation for matching entities
minStdDevStep	5	minimum standard deviation step size
eblaLoopCount	5	number of times to process all experiences for each minimum standard deviation
fixedStdDev	false	Boolean flag indicating whether to limit standard deviation to value specified
tmpPath	./ebla/	temporary path for processing movies (experiences)
displayMovie	false	Boolean flag indicating whether to display movies while extracting frames
displayText	false	Boolean flag indicating whether to generate/display detailed intermediate results while processing
saveImages	true	Boolean flag indicating whether to save movie frames after processing/analysis
segColorRadius	6.5	float containing the color radius for mean shift analysis image segmentation
segSpatialRadius	7	integer containing the spatial radius for mean shift analysis image segmentation
segMinRegion	20	integer containing the minimum number of pixels that constitute a region for mean shift analysis image segmentation
segSpeedUp	1	integer containing the speed-up level for mean shift analysis image segmentation (0=none, 1=medium, 2=high)
framePrefix	/frame	string containing file prefix for temp frames extracted from each movie (experience)
segPrefix	/seg	string containing file prefix for temp segmented images created for each frame
polyPrefix	/poly	string containing file prefix for temp polygon images created for each frame
backgroundPixels	20.0	float containing the percentage of total pixels that an object must contain to be considered part of the background rather than a significant object (0 - 100)
minPixelCount	500	integer containing the minimum number of pixels that constitute a significant object
minFrameCount	7	integer containing the minimum number of consecutive frames that an object must appear in to be considered a significant object (helps to eliminate noise and shadows).
reduceColor	false	Boolean flag indicating whether to reduce the color depth of any segmented regions
caseSensitive	false	Boolean flag indicating whether lexemes are case sensitive
notes	(blank)	string containing notes about current set of runtime parameters

5.6 – Video Processing

Once the runtime parameters have been retrieved, the *processExperiences()* method of EBLA is called to retrieve a list of experiences from the **experience_data** table of the database. It loops through the experience records, calling the top-level classes necessary to process each.

The first class called by the *processExperiences()* method of the *EBLA* class is *FrameGrabber*. Whether animations or actual videos, experiences are presented to EBLA as short movies. The *FrameGrabber* class extracts the individual frames from these movies and saves them as individual graphics files.

FrameGrabber takes a source video file from the **experience_data** table and an output path from *Params* and then uses classes from the Java Media Framework (JMF) API to process the movie. The Java Media Framework (JMF) API contains classes for playing and manipulating several common video formats including Audio Video Interleave (AVI) and QuickTime Movie (MOV). As *FrameGrabber* processes a video file, each frame is extracted or “ripped,” converted to a *BufferedImage* object, and saved using the Portable Network Graphics (PNG) graphics format. The *BufferedImage* class is a standard part of the Java language used for image handling, and the PNG graphics format is an open specification that features lossless compression and is supported on most platforms. The image files produced by *FrameGrabber* are indexed and padded with leading zeros so that they can be viewed alphabetically using most graphics software packages. Additional features of *FrameGrabber* include a *main()* routine for standalone execution and an option to display or hide each movie that it processes.

As a final note, when *FrameGrabber* was first implemented, it was designed to output JPEG rather than PNG files. While the JPEG format can produce smaller files, its lossy

compression technique caused some problems for the vision system discussed later in this chapter.

5.7 – Frame Processing

The second class called for each experience is *FrameProcessor*. *FrameProcessor* takes the PNG files generated by *FrameGrabber* and analyzes their content for significant objects and relationships among those objects. Based on the approach taken in Siskind's most recent work with event recognition, *FrameProcessor* begins by calculating a bounding polygon for each significant object in every frame. (Siskind 2001) These polygons are then used to calculate object attributes including area, centroid (center of gravity), and position. Unfortunately, calculating an accurate bounding polygon is not a simple task, and two quite different versions of *FrameProcessor* were written before an acceptable result was obtained.

5.7.1 – Edge Detection

The first version of *FrameProcessor* developed for EBLA calculated bounding polygons by loading each image, performing average pixel filtering, converting the image to grayscale, and thresholding the image to black and white using a comparison to the average grayscale pixel value. It then performed Laplacian-Prewitt edge detection and traversed the remaining single-pixel edges forming polygon definitions from the line segments. Several other edge detection methods were evaluated, including Roberts and Sobel, but the Laplacian-Prewitt method typically left only single pixel edges, making the traversal quite efficient. (Hardy 2000; Lyon 1999)

Although it was fast, the combination of grayscale conversion, thresholding, and edge detection had several shortcomings. First, converting to grayscale removed color information from the images. Second, thresholding about the average grayscale value had a tendency to discard some

or all of a significant object and/or include noise and shadows. Finally, edge detection outlined multiple objects as a single object if there was any contact among objects. Figure 16 shows the original images and polygon traces for several frames from an early experience. Based on the problems with the preliminary results, most of the original code for *FrameProcessor* was abandoned.

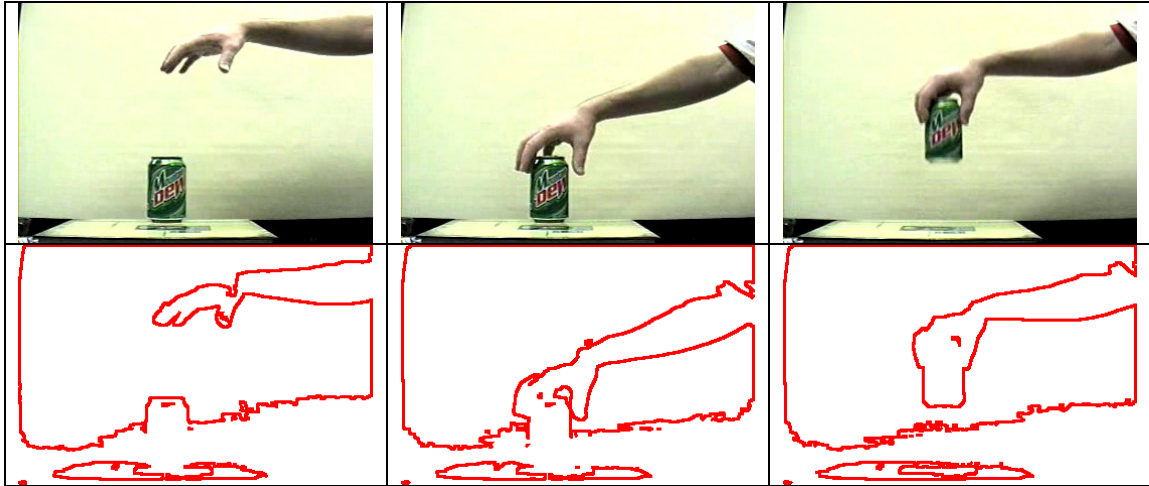


Figure 16. Video Frames and Polygon Tracings from Original Edge Detector

5.7.2 – Image Segmentation

The revised *FrameProcessor* detects significant objects using a robust color segmentation algorithm based on mean shift analysis. (Comaniciu and Meer 1997; 1999; Comaniciu 2002) In general terms, color image segmentation involves identifying the clusters of significant features in an image. Mean shift analysis is a nonparametric feature space analysis technique that provides an estimate of the density gradient for each point in a given feature space. In more formal terms:

Let $f(\mathbf{x})$ be the (unknown) probability density function underlying a p -dimensional feature space, and \mathbf{x}_i the available data points in this space. Under its simplest formulation, the mean shift property can be written as

$$\widehat{\nabla f(\mathbf{x})} \sim \left(\text{ave}_{\mathbf{x}_i \in S_{h,\mathbf{x}}} [\mathbf{x}_i] - \mathbf{x} \right) \quad (1)$$

where $S_{h,x}$ is the p -dimensional hypersphere with radius h centered on x . Relation (1) states that the estimate of the density gradient at location x is proportional to the offset of the mean vector computed in a window, from the center of that window. (Christoudias, Georgescu, and Meer 2002, 2)

When the mean shift estimate is applied recursively, it is guaranteed to converge on local density maxima. These maxima can then be used to cluster the feature space.

For the specific task of color image segmentation, the feature space consists of a two-dimensional spatial domain containing pixel locations and a three-dimensional range domain containing the color values of each pixel. Since the common RGB color space is not perceptually uniform, each pixel value is nonlinearly transformed to the L^*u^*v color space. The mean shift algorithm requires three parameters, h_s , h_r , and M where h_s governs the spatial resolution, h_r governs the color (range) resolution and M determines the minimum number of pixels that constitute a significant region. (Comaniciu and Meer 1997; 1999; Comaniciu 2002)

The mean shift segmentation algorithm proceeds as follows:

Let x_i and z_i , $i = 1, \dots, n$, be the d -dimensional input and filtered image pixels in the joint spatial-range domain and L_i the label of the i th pixel in the segmented image.

1. Run the mean shift filtering procedure for the image and store *all* the information about the d -dimensional convergence point in z_i , i.e., $z_i = y_{i,c}$.
2. Delineate in the joint domain the clusters $\{C_p\}_{p=1..m}$ by grouping together all z_i which are closer than h_s in the spatial domain and h_r in the range domain, i.e., concatenate the basins of attraction of the corresponding convergence points.
3. For each $i = 1, \dots, n$, assign $L_i = \{p \mid z_i \in C_p\}$.
4. Optional: Eliminate spatial regions containing less than M pixels. (Comaniciu 2002, 11)

Although significantly slower than the edge detection approaches, the mean shift algorithm does an excellent job of detecting significant objects and reducing each one to a single color value. This approach has the benefits of retaining color information and separating objects that are in contact with one another.

The initial implementation of the mean shift algorithm was based on a prototype Java application written by Comaniciu and Meer. The segmentation classes from the application were added to the *com.greatmindsworking.EBLA* package and modified to accept graphics files in the Portable Network Graphics (PNG) format. *FrameProcessor* was retooled to instantiate the segmentation classes, which would then generate a list of the pixels for each segmented region. *FrameProcessor* would then pass these lists to several internal post-processing methods responsible for removing the interior pixels from each region and traversing the remaining borders to form polygon definitions.

Initial results were promising. While requiring some tuning for each series of movies, a single set of mean shift parameter values would properly segment about 95% of each movie's frames. The initial test set of movies had quite a bit of noise due to glare, shadows, and blurring in action frames, and it was thought that filming under better conditions with a digital camcorder would lead to even better segmentation results.

Unfortunately, even with movies of significantly higher quality, the segmentation routines continued to undersegment or oversegment a few frames in each movie, either combining multiple objects into a single region or breaking single objects into multiple regions. In addition, having to fine tune the segmentation parameters for different sets of movies meant that the EBLA Model would require more human intervention than was desirable. Figure 17 shows examples of oversegmentation and undersegmentation.

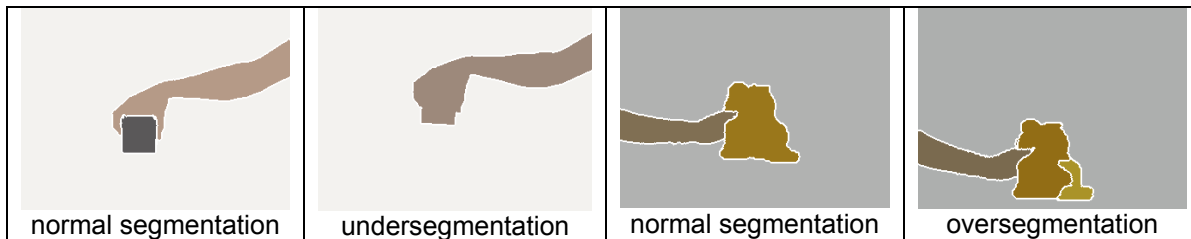


Figure 17. Normal Segmentation, Oversegmentation, and Undersegmentation

To overcome this, either the algorithm had to be improved, or the real videos had to be discarded in lieu of simpler animations with solid-colored regions and little or no noise. Fortunately, Christoudias, Georgescu, and Meer at Rutgers were working on an enhanced version of the segmentation algorithm as part of a vision toolkit called EDISON. The revised algorithm allowed for additional parameter tuning, contained many fixes and optimizations, and was thoroughly documented. The only downside was that it was written entirely in C++ and had to be ported to Java in order to operate seamlessly with EBLA. (Christoudias, Georgescu, and Meer 2002)

The segmentation code from EDISON was ported into a new package called *com.greatmindsworking.EDISON.segm*. This allows Java programmers interested in image segmentation to make use of the ported code outside of EBLA. While porting the segmentation libraries in EDISON was a significant undertaking involving over 9,000 lines of code, the results were well worth the effort. The new code is significantly faster than the original prototype code, and it can correctly segment an entire series of videos without adjustment of the segmentation parameters.

Upon completion of the port, *FrameProcessor* was adjusted to instantiate the classes in the new EDISON package. All of the parameters needed by the segmentation classes were added to the *Params* class along with two threshold values for eliminating background regions and very small pixel regions. The threshold for eliminating the frame background as a significant region is specified as a percentage of the total number of pixels in the frame. The threshold for eliminating small regions is specified as a simple pixel count.

As a final measure to clean up and modularize *FrameProcessor*, the algorithms to process the segmented regions detected by EDISON were improved and moved into a separate

class named *RegionTracer*. *RegionTracer* takes a list of the pixel coordinates for every segmented region and forms a bounding polygon for each. It begins by removing any outlier pixels having less than three neighboring pixels. It then picks the left-most pixel in the top row and begins to trace the edge pixels in a counter-clockwise fashion based on a global orientation and a local heuristic.

RegionTracer's global orientation is initially set to “south” and from each point it attempts to travel “left,” “forward,” and “right.” For any direction other than forward, *RegionTracer* adjusts its global orientation to “north,” “south,” “east,” or “west” based on the current global orientation and the new local direction. As *RegionTracer* traverses each pixel, it changes that pixel’s value to reflect its current global orientation. This allows it to backtrack if, at any point, it is unable to make forward progress. *RegionTracer* uses the absolute distance from the current position to the starting position as its stopping criteria. If that distance is less than a certain threshold (currently five pixels), the trace terminates. To prevent the trace from terminating prematurely, *RegionTracer* must successfully traverse a minimum number of pixels (currently seventy). If all points are exhausted during a search and the stopping criteria are not met, *RegionTracer* terminates and *FrameProcessor* drops the current frame.

5.7.3 – Polygon Analysis

Using the polygon list for each frame generated by *RegionTracer*, an initial analysis of the frame is performed. *FrameProcessor* instantiates a class named *FrameObject*, determines the area, centroid, number of polygon vertices, and bounding rectangle for each significant object. The first point in each polygon is then used to obtain the average RGB color value for each region in the segmented image. Based the *reduceColor* Boolean variable in *Params*, this RGB value can then be reduced from sixteen million possible values to twenty-seven.

In order to track objects across multiple frames, a correlation index is calculated for every object. In the first frame, objects are numbered from the top-left to the bottom-right. In subsequent frames, a normalized score is calculated for every possible object pairing using differences in the X and Y coordinates of the centroid along with differences in area. These scores are ranked using a class called *MatchScore*. Pairings are established based on the lowest score, the newly paired objects are removed from the list of possible pairings for the prior and current frames, and the process is repeated until all object correlations are established. Note that if an object drops out of sight in any frame and then returns, it is treated as a new object because the correlation process only performs a single frame look-back.

5.7.4 – Intermediate Results

The results of the polygon analysis are written to the **frame_analysis_data** table in the **ebla_data** database. The rest of the processing done by the EBLA system is based on the data in this table. Not only does this allow for external analysis of intermediate results, but also allows the intensive vision processing portions of EBLA to be run separately from the rest of the entity and lexeme processing. In fact, the vision pre-processing modules, the vision post-processing modules, and the language resolution modules can all be executed independently, based on the *processVideos*, *processEntities*, and *processLexemes* Boolean variables in *Params*. This feature is particularly useful when testing and debugging EBLA.

Params contains another Boolean variable, *saveImages*, which allows the user to visually inspect the results of the preliminary frame processing. *FrameProcessor* can generate two intermediate PNG files for each frame. Both files are placed in the same output path used by *FrameGrabber*. The first file is prefixed with “seg” by default and contains the segmented version of each frame. The second file is prefixed with “poly” by default and contains a colored

polygon representation of each frame over a white background. The centroid and bounding rectangle for each polygon are drawn as black and red boxes respectively. The file name prefixes are contained in the *segPrefix* and *polyPrefix* variables in *Params* and can be adjusted in the **parameter_data** table. Sample segmented images are shown in figure 18 and sample polygon images are shown in figure 19.

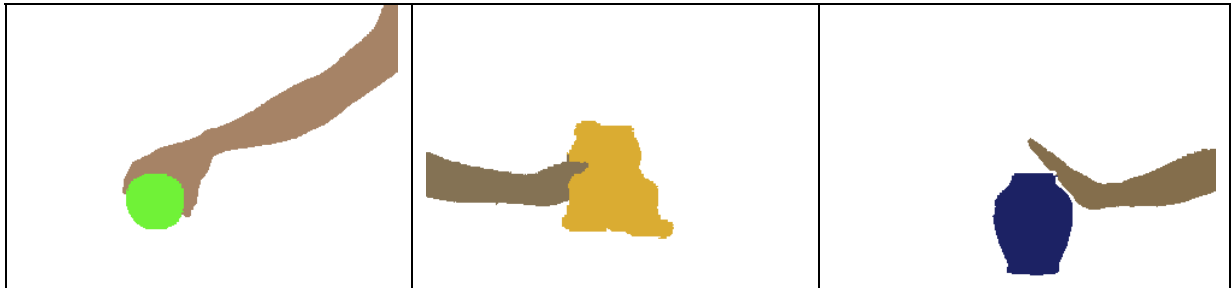


Figure 18. Sample Segmented Images

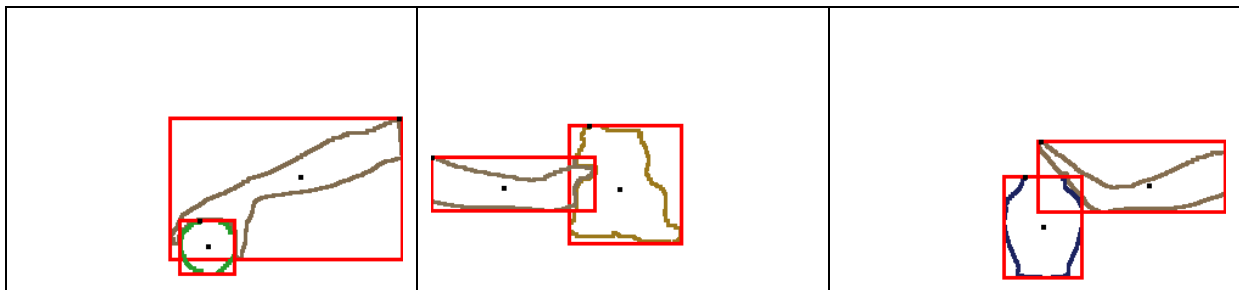


Figure 19. Sample Polygon Images

5.8 – Entity Extraction

Once all frames for an experience have been extracted and preprocessed, EBLA instantiates the *EntityExtractor* class to perform video post-processing. *EntityExtractor* analyzes the intermediate results in the **frame_analysis_data** table to identify the objects and relationships across all frames. When EBLA was initially designed, it was thought that objects and object-object relations would have to be stored separately due to the abstract nature of the dynamic relationships among objects. Upon further consideration, an entity-attribute model was designed based loosely on the linking feature structures (f-structs) used in Bailey’s work.

(Bailey 1997) The basic idea is to treat both objects and relationships among objects as entities. In order to do this, a set of attributes is established to define and describe all possible entities. An entity is then stored as a set of attribute-value pairs in which each value is an average over all of the frames in a given experience.

In the **ebla_data** database, the entity-attribute system is implemented using four tables. The first, **attribute_list_data** contains a list of all object and relation attributes calculated in *EntityExtractor*. Currently, these calculations are hard-coded, but the **attribute_list_data** table is designed with a field for specifying a Java class name for each attribute. This field could one day be used in conjunction with the Java *forName()* method to dynamically load attribute calculations. This would allow users to specify their own attributes and corresponding calculations without recompiling the entire EBLA system. Fields have also been provided in **attribute_list_data** to disable attributes and to specify whether a given attribute should be applied to individual objects or to object-object relations.

The second table, **entity_data**, contains only a unique entity identifier and an occurrence count to track how many times a given entity has been encountered while processing experiences. The third table, **attribute_value_data**, links **attribute_list_data** and **entity_data**, and it contains one record for each attribute-value pair defining an entity. The final table, **experience_entity_data**, links **entity_data** to **experience_data** and contains one record for each entity in a given experience. This table also contains a field that indicates if each instance of an entity has been matched to a lexeme.

In general terms, *EntityExtractor* operates by calculating the object attribute values for each object and the relation attributes for each pair of objects in every frame of an experience. The data for each attribute is stored in a Java *ArrayList* data structure and passed to the

ArrayAnalysis class. The *ArrayAnalysis* class calculates statistical information for each *ArrayList* including average, standard deviation, minimum, and maximum.

The results for each potential entity are compared to values for existing entities in the database. If a match is found, the occurrence count for that entity is updated, and its attribute values in **attribute_value_data** are updated to reflect the latest occurrence using a weighted average. In the event that multiple matches are returned, the entities are ranked based on how closely the attribute values for each entity match the attribute values for the potential entity, and the closest entity is chosen. If a match is not found, new records are added to **entity_data** and **attribute_value_data** to reflect the new entity. In either case, a new record is added to **experience_entity_data**.

A matching entity must have average values for *all* attributes within a single standard deviation (σ) of the averages for the current object. When σ for an attribute is less than a specified percentage of the average, the specified percentage is used instead. This minimum standard deviation (σ_{\min}) determines how much two entities must differ to be considered distinct, and thus can have a significant impact on the number of unique entities recognized by EBLA. A wide range of values for σ_{\min} were used to evaluate EBLA, and both the testing procedure and the results are discussed in the next chapter.

5.8.1 – Object Entities

In order for an object to be considered an entity, it must exist in some minimum number of consecutive frames. This parameter is contained in the *minFrameCount* variable of the *Params* class and is set once for all experiences. Establishing this minimum number of frames helps to prevent phantom objects, caused by noise, from distorting the results.

There are five attributes calculated for each object. The first two, area and number of polygon edges, come directly from the results of the *FrameProcessor* class. The third, grayscale value, ranges from 0 to 255 and is calculated by averaging the red, green, and blue components of the object's color. The remaining two attributes represent the horizontal and vertical coordinates of the object's centroid relative to the width and height respectively of the object's bounding rectangle. In more formal terms, the horizontal and vertical values of the relative centroid, RC_x and RC_y , can be defined as follows:

1. Let C_x and C_y be the respective horizontal and vertical coordinates of the object's centroid.
2. Let BR_x and BR_y be the respective rightmost horizontal and bottommost vertical coordinates of the object's bounding rectangle.
3. Let BR_w and BR_h be the respective width and height of the object's bounding rectangle.
4. Set $RC_x \leftarrow (C_x - BR_x) / BR_w$.
5. Set $RC_y \leftarrow (C_y - BR_y) / BR_h$.

The relative centroid provides abstracted information about the shape and uniformity of an object. By dividing by the width and height of the bounding rectangle, the values are normalized so that they will always range from zero to one and will not vary based on the size of the object.

5.8.2 – Relation Entities

For purposes of EBLA, two objects are considered to have a significant relationship if they come in contact for at least one frame of a given experience. Contact is assumed to occur for object pairs when their bounding rectangles overlap. Once contact is detected, a number of spatial relation attributes are measured for the object pair over the entire experience. Other than

contact, all of the spatial relations are based on changes in the position of the centroid from frame to frame.

The set of relation attributes calculated by *EntityExtractor* was determined experimentally, based in part on the feature vectors used by Siskind in his HOWARD system. (Siskind 1999) The current relation attributes for EBLA are shown in table 5.

Table 5. Relation Attributes for EBLA

Attribute	Calculation
contact	set to 0 if bounding rectangles of polygons overlap set to 1 if bounding rectangles of polygons do not overlap
x relation	set to -1 if object #1's centroid is left (west) of object #2's centroid set to 0 if object #1's centroid has same x-coordinate as object #2's centroid set to 1 if object #1's centroid is right (east) of object #2's centroid
y relation	set to -1 if object #1's centroid is over (north of) object #2's centroid set to 0 if object #1's centroid has the same y-coordinate as object #2's centroid set to 1 if object #1's centroid is below (south of) object #2's centroid
delta x	set to -1 if horizontal distance between centroids is decreasing set to 0 if horizontal distance between centroids is unchanged set to 1 if horizontal distance between centroids is increasing
delta y	set to -1 if vertical distance between centroids is decreasing set to 0 if vertical distance between centroids is unchanged set to 1 if vertical distance between centroids is increasing
x-travel	add 1 for each object moving right and subtract 1 for each object moving left
y-travel	add 1 for each object moving down and subtract 1 for each object moving up

They operate on a point system based on spatial relations. By design, the attributes do not capture the magnitude of any relation. Rather, the goal is to describe the essence of a spatial relation over time. For example, if one object picks up another, it is not important how far apart the objects are or how quickly the first object approaches the second. What is important is that the first object approaches the second, establishes contact, and remains in contact as both objects travel vertically.

5.9 – Lexical Resolution

The final, and perhaps most important, class instantiated by EBLA is *LexemeResolver*. *LexemeResolver* is responsible for parsing the protolanguage description of each event from the

experience_data table and resolving the individual lexemes in that description to their corresponding entities.

5.9.1 – Parsing

The parsing logic phase of *LexemeResolver* begins by extracting the individual lexemes (words) for the current experience from the protolanguage description in the **experience_data** table of the **ebla_data** database. The description is passed to *LexemeResolver* as a space-delimited string and tokenized using the Java *StringTokenizer* class. Note that *StringTokenizer* is case-sensitive so “ball,” “Ball,” and “BALL” are all considered to be distinct tokens. This case-sensitivity is overridden by default based on the *caseSensitive* Boolean variable in the *Params* class. When *caseSensitive* is set to false, the *processExperiences()* method of *EBLA* simply converts the description to all lowercase characters using the *toLowerCase()* method of the *String* class before passing it to *LexemeResolver*.

After *LexemeResolver* tokenizes the description, the lexemes are compared to the existing lexemes in the **lexeme_data** table. Each record in the **lexeme_data** table contains a unique identifier, a lexeme, and an occurrence count. For the current experience, if a match is found, the occurrence count is incremented. Otherwise, a new record is added with an occurrence count of one. In either case, a record is added to the **experience_lexeme_data** table, which maps **experience_data** to **lexeme_data** and indicates whether or not a lexeme has been resolved for each experience. Initially all **experience_lexeme_data** records are assumed to be unresolved.

5.9.2 – Cross-Situational Learning

Once the lexemes for a given experience have been tokenized and added to the database, *LexemeResolver* retrieves all of the entities for that experience and begins the actual lexical resolution process. For each experience, two Java *ArrayLists* are created to hold all of the

unresolved entities and lexemes. As each entity-lexeme pairing is resolved, a record is added to the **entity_lexeme_data** table with the entity and lexeme unique identifiers and an occurrence count. As new examples of an existing mapping are encountered, the occurrence count for that mapping is incremented. The following general procedure for resolving entity-lexemes mappings was listed in the previous chapter, but is provided again here for convenience:

1. Lookup any known resolutions from prior experiences.
2. Resolve any single remaining entity-lexeme pairings.
3. Apply cross-situational inference, comparing unresolved entities and lexemes across all prior experiences, repeating stage two after each new resolution.

To perform the first stage of lexical resolution, *LexemeResolver* queries the **entity_lexeme_data** table for any records with both an entity and lexeme from the current experience. If any records are returned, those pairings are removed from the unresolved entity and lexeme *ArrayLists* and the resolution indicators are updated in the **experience_entity_data** and **experience_lexeme_data** tables. The occurrence count in the **entity_lexeme_data** table is incremented accordingly.

The second stage simply involves checking the number of remaining items in the unresolved entity and lexeme *ArrayLists*. If both lists contain only a single item, then those items are removed from their respective lists and are used to produce a new entity-lexeme mapping. A new **entity_lexeme_data** record is created, and the resolution indicators are updated in the **experience_entity_data** and **experience_lexeme_data** tables. Prior experiences are then searched for additional instances of the same unresolved pairing. If any are discovered, they are resolved accordingly, and the entire process is repeated until no new resolutions are made.

As discussed in section 4.7, the third stage involves applying a cross-situational inference technique and is the most complex from a computational standpoint. It requires calculating the set intersections and differences for all of the unresolved entities and lexemes across all experiences in a pair wise fashion. These calculations are implemented in the *resolveLexemes()* method of *LexemeResolver* using the database queries shown in figure 20, figure 21, and figure 22.

Since the queries for the third stage are not very efficient in comparison to those for the second stage, they are limited to a single result. If a new mapping is discovered, the second stage of *LexemeResolver* is repeated to detect the more obvious mappings before any additional stage three queries are executed. Stages two and three are applied to the database repeatedly until no additional mappings are found.

```
// TECHNIQUE #1 - FIND ALL UNMATCHED SETS WITH SINGLE OVERLAP & SOLVE
// RETURN ONLY FIRST RESULT
// ID1 < ID2 (DOESN'T REALLY MATTER FOR THIS CASE) AND MATCH COUNT = 1

    eedSQL = "(SELECT eed1.experience_id AS id1, eed2.experience_id AS id2, COUNT(*)"
      + " FROM experience_entity_data eed1, experience_entity_data eed2"
      + " WHERE (eed1.resolution_code = 0)"
      + " AND (eed2.resolution_code = 0)"
      + " AND (eed1.entity_id = eed2.entity_id)"
      + " AND (eed1.experience_id < eed2.experience_id)"
      + " GROUP BY eed1.experience_id, eed2.experience_id"
      + " HAVING COUNT(*) = 1)";

    eldSQL = "(SELECT eld1.experience_id AS id1, eld2.experience_id AS id2, COUNT(*)"
      + " FROM experience_lexeme_data eld1, experience_lexeme_data eld2"
      + " WHERE (eld1.resolution_code = 0)"
      + " AND (eld2.resolution_code = 0)"
      + " AND (eld1.lexeme_id = eld2.lexeme_id)"
      + " AND (eld1.experience_id < eld2.experience_id)"
      + " GROUP BY eld1.experience_id, eld2.experience_id"
      + " HAVING COUNT(*) = 1)";

    sql = "SELECT * FROM " + eedSQL + " e, " + eldSQL + " l"
      + " WHERE e.id1 = l.id1"
      + " AND e.id2 = l.id2"
      + " LIMIT 1;";
```

Figure 20. Set Intersection Query for Lexical Resolution

```

// TECHNIQUE #2 - FIND ALL MATCHED SETS WHERE
// #MATCHES = (#UNRESOLVED INSTANCES IN FIRST EXPERIENCE-1) AND (EXP_ID_1 < EXP_ID_2)
// RETURN ONLY FIRST RESULT
// ID1 < ID2 AND MATCH COUNT = # UNRESOLVED ENTITIES/LEXEMES IN FIRST EXPERIENCE - 1
// TO PROPERLY HANDLE THE SITUATION WHERE AN ENTITY EXISTS TWICE IN AN EXPERIENCE
// (DUE TO IT REALLY EXISTING TWICE OR OVERGENERALIZATION (E.G. BALL AND HAND MAP
// TO SAME ENTITY)), ONLY DISTINCT PAIRINGS ARE TAKEN FROM THE "B" SET
eedSQL = "(SELECT eed1.experience_id AS id1, eed2.experience_id AS id2, COUNT(*) as ecnt"
+ " FROM experience_entity_data eed1, "
+ " (SELECT DISTINCT experience_id, entity_id FROM experience_entity_data
WHERE resolution_code=0) eed2"
+ " WHERE (eed1.resolution_code = 0)"
+ " AND (eed1.entity_id = eed2.entity_id)"
+ " AND (eed1.experience_id < eed2.experience_id)"
+ " GROUP BY eed1.experience_id, eed2.experience_id"
+ " HAVING COUNT(*) = ((SELECT COUNT(*) FROM experience_entity_data"
+ " WHERE experience_id = eed1.experience_id AND resolution_code=0) - 1));"

eldSQL = "(SELECT eld1.experience_id AS id1, eld2.experience_id AS id2, COUNT(*) as lcnt"
+ " FROM experience_lexeme_data eld1, "
+ " (SELECT DISTINCT experience_id, lexeme_id FROM experience_lexeme_data
WHERE resolution_code=0) eld2"
+ " WHERE (eld1.resolution_code = 0)"
+ " AND (eld1.lexeme_id = eld2.lexeme_id)"
+ " AND (eld1.experience_id < eld2.experience_id)"
+ " GROUP BY eld1.experience_id, eld2.experience_id"
+ " HAVING COUNT(*) = ((SELECT COUNT(*) FROM experience_lexeme_data"
+ " WHERE experience_id = eld1.experience_id AND resolution_code=0) - 1));"

sql = "SELECT * FROM " + eedSQL + " e, " + eldSQL + " l"
+ " WHERE e.id1 = l.id1"
+ " AND e.id2 = l.id2"
+ " LIMIT 1;";

```

Figure 21. First Set Difference Query for Lexical Resolution

```

// TECHNIQUE #3 - FIND ALL MATCHED SETS WHERE
// #MATCHES = (#UNRESOLVED INSTANCES IN FIRST EXPERIENCE-1) AND (EXP_ID_1 > EXP_ID_2)
// RETURN ONLY FIRST RESULT
// ID1 > ID2 AND MATCH COUNT = # UNRESOLVED ENTITIES/LEXEMES IN FIRST EXPERIENCE - 1
// TO PROPERLY HANDLE THE SITUATION WHERE AN ENTITY EXISTS TWICE IN AN EXPERIENCE
// (DUE TO IT REALLY EXISTING TWICE OR OVERGENERALIZATION (E.G. BALL AND HAND MAP
// TO SAME ENTITY)), ONLY DISTINCT PAIRINGS ARE TAKEN FROM THE "B" SET
eedSQL = "(SELECT eed1.experience_id AS id1, eed2.experience_id AS id2, COUNT(*)"
+ " FROM experience_entity_data eed1, "
+ " (SELECT DISTINCT experience_id, entity_id FROM experience_entity_data
WHERE resolution_code=0) eed2"
+ " WHERE (eed1.resolution_code = 0)"
+ " AND (eed1.entity_id = eed2.entity_id)"
+ " AND (eed1.experience_id > eed2.experience_id)"
+ " GROUP BY eed1.experience_id, eed2.experience_id"
+ " HAVING COUNT(*) = ((SELECT COUNT(*) FROM experience_entity_data"
+ " WHERE experience_id = eed1.experience_id AND resolution_code=0) - 1));"

eldSQL = "(SELECT eld1.experience_id AS id1, eld2.experience_id AS id2, COUNT(*)"
+ " FROM experience_lexeme_data eld1, "
+ " (SELECT DISTINCT experience_id, lexeme_id FROM experience_lexeme_data
WHERE resolution_code=0) eld2"
+ " WHERE (eld1.resolution_code = 0)"
+ " AND (eld1.lexeme_id = eld2.lexeme_id)"
+ " AND (eld1.experience_id > eld2.experience_id)"
+ " GROUP BY eld1.experience_id, eld2.experience_id"
+ " HAVING COUNT(*) = ((SELECT COUNT(*) FROM experience_lexeme_data"
+ " WHERE experience_id = eld1.experience_id AND resolution_code=0) - 1));"

sql = "SELECT * FROM " + eedSQL + " e, " + eldSQL + " l"
+ " WHERE e.id1 = l.id1"
+ " AND e.id2 = l.id2"
+ " LIMIT 1;";

```

Figure 22. Second Set Difference Query for Lexical Resolution

LexemeResolver can be executed in both a “resolution mode” and a “description” mode. In resolution mode, the *resolveLexemes()* method of *LexemeResolver* attempts to resolve the lexemes in the protolanguage description for the current experience using the three steps described above. In test mode, the *generateDescriptions()* method of *LexemeResolver* attempts to generate a protolanguage description of the experience based on entity-lexeme mappings from prior experiences. In this mode, if an entity has not been processed in a prior experience, *LexemeResolver* will insert “[unknown]” as a placeholder. The *generateDescriptions()* method can score itself if a correct description is provided. It tracks the number of correct, incorrect, and unknown lexemes generated. If an entity has multiple lexeme mappings, all are processed and scored. For example if *generateDescriptions()* was processing “hand pickup ring,” and the object entity that matched the ring had been mapped to “ring” and “bowl,” both the correct and incorrect lexeme counts would be incremented.

5.10 – Summary

This chapter has provided a technical discussion of the software implementation for the EBLA Model. EBLA is a complete software system that has been constructed using object-oriented principles and successfully tested on multiple platforms. It has been designed with future research in mind and should be straightforward to extend.

The next chapter outlines the criteria used to evaluate EBLA and the procedures used to generate experiences for EBLA. In addition, it presents and summarizes the experimental results.

CHAPTER 6 – EVALUATION OF THE EBLA MODEL

6.1 – Evaluation Criteria

There are several criteria that can be used to measure the lexical acquisition performance of EBLA. First, overall success can be evaluated by determining how many correct entity-lexeme mappings are generated in comparison to the total number of entities detected. Second, acquisition speed can be measured by looking at the average number of experiences needed to resolve a word in comparison to the total number of experiences processed. Third, robustness can be measured by presenting EBLA with new, unlabeled experiences and measuring its ability to generate protolanguage descriptions based on prior experiences.

The performance of the entity detection system will be discussed in casual terms, but no attempt has been made to measure it explicitly. The attributes used to define object and relation entities were established experimentally and are quite subjective in nature. For example, the hand used throughout the real videos produced for EBLA varied in size, color, orientation, and form depending on the type of event taking place. Based on this variation in context, EBLA created several hand entities in the database. While the number of hand entities detected could easily be changed by enabling, disabling, or changing object attributes, the ideal number of hand entities is debatable and really a matter of personal taste. Model-merging will be discussed in chapter 7 as a possible way to extend EBLA to consolidate, and thus decontextualize, similar entity definitions.

6.2 – Creation of Animations

EBLA can process both animations and actual videos as long as they are in the Audio Video Interleave (AVI) or QuickTime Movie (MOV) video formats supported by the Java Media Framework. The first experiences created for EBLA were a set of eight simple animations with

consistent coloring, shape, movement, size, and lighting. These simplified experiences greatly aided the testing and debugging process because EBLA’s calculation results were much more predictable. The animations were created using Macromedia Flash and were designed using solid polygon drawings of an arm, a hand, a green ball, and a red box. The arm and hand were combined with each object to create *pickup*, *putdown*, *touch*, and *slide* animations. These animations were then exported as AVI files so that they could be delivered to EBLA in the same manner that actual videos would be delivered.

6.3 – Evaluation of Animations

Due to their uniform colorings and clearly defined edges, the animations were processed successfully by the vision system using the default segmentation parameters of $h_s = 7$, $h_r = 6.5$, $M = 20$, and the medium speedup option. EBLA created single object entities for the green ball and red box and two object entities for the arm/hand. One arm/hand entity was generated for the *pickup* and *putdown* events and another for the *touch* and *slide* events. This was as expected based on the differing hand shapes (see figure 23).

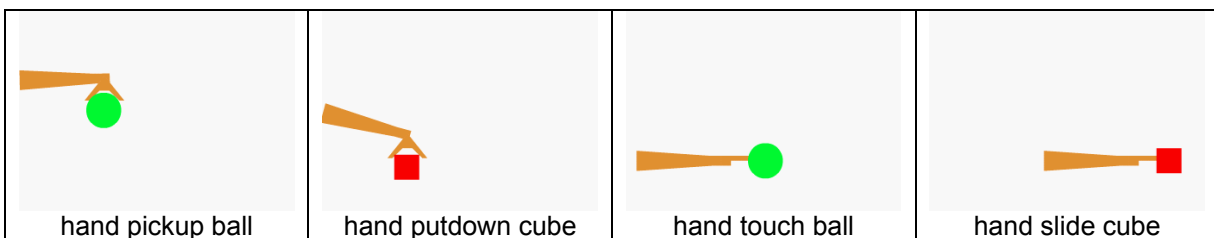


Figure 23. Variation in Shape of Arm/Hand Used in Animations

When EBLA was first tested using the animations, only five of the seven relation attributes had been added to the **attribute_list_data** table in the **ebla_data** database. EBLA correctly generated distinct relation entities for *pickup* and *putdown*, but lumped *touch* and *slide* into a single entity. The “x-travel” and “y-travel” attributes discussed in sections 4.6 and 5.8.2 were added to the model to rectify this.

The eight animation experiences were delivered to EBLA at random, ten times for each of nineteen different minimum standard deviation (σ_{\min}) values, yielding a total of 190 runs. The value of σ_{\min} used to match the attribute values to existing entities was varied from 5% to 95% in increments of 5%. Figure 24 shows the average acquisition time (in number of experiences) based on the number of experiences processed. A third-order polynomial curve was fit to the data. What this graph indicates is that for the first of the eight experiences, an average of three experiences was required to resolve all of the entity-lexeme mappings. From the fifth entity on, most entity-lexeme mappings were already established and were therefore resolved immediately. Depending on the order in which the animations were processed, some entity-lexeme mappings were established in as few as two experiences.

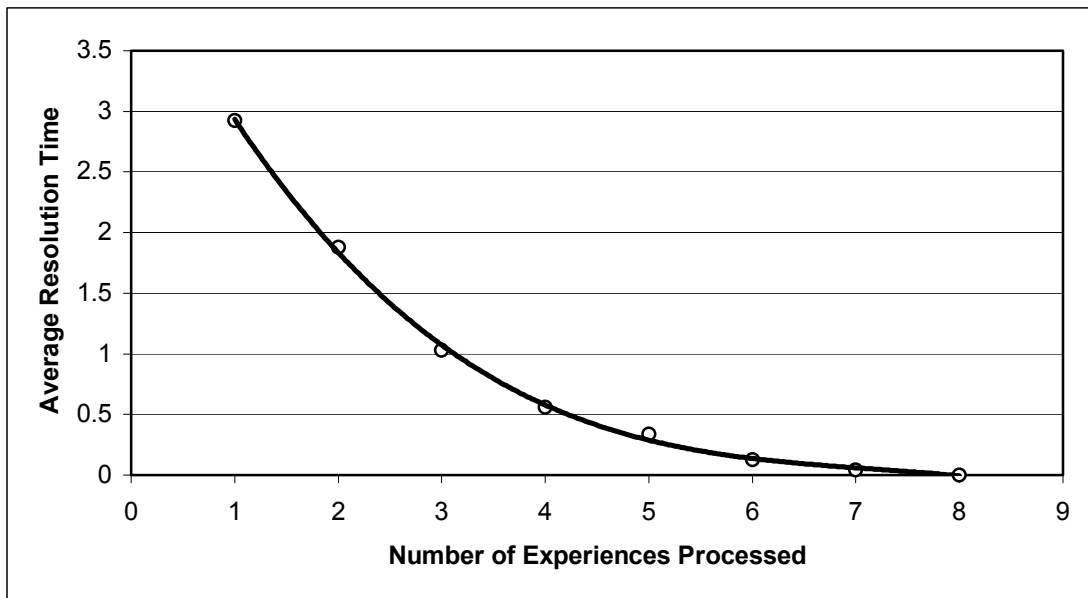


Figure 24. Average Lexical Acquisition Time for Animations

Figure 25 shows the success rates for lexeme mappings for each of the nineteen σ_{\min} values. When σ_{\min} was less than 40%, EBLA was able to resolve all of the lexemes and entities. For σ_{\min} values greater than 40%, the lexeme resolution rate tapered off, and the rate dropped just below 80% for σ_{\min} values of 85% and 95%. This drop off can be attributed to the

overgeneralization of entity definitions that occurs for higher σ_{\min} values. Similar entities such as the *touch* and *slide* relation entities get lumped into a single definition, and EBLA is unable to overcome the additional referential ambiguity.

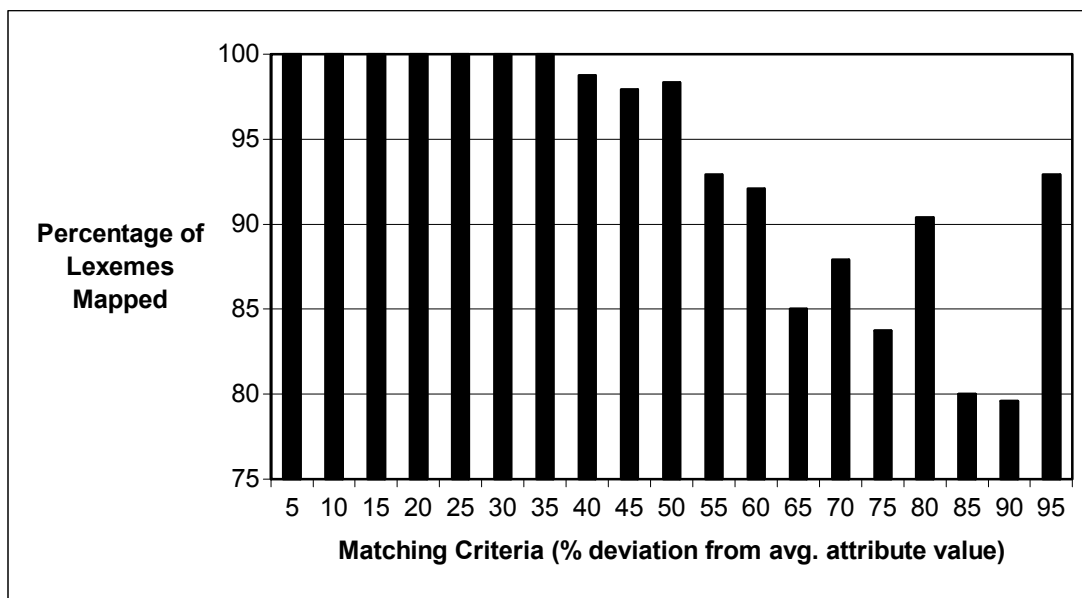


Figure 25. Lexeme Mapping Success Rates for Different Minimum Standard Deviations

The animation test set was also used to evaluate EBLA’s ability to produce descriptions for novel experiences. Since there was little variation in the entities for the animations, all of the description mode runs used a σ_{\min} value of 5%. For the first run, EBLA randomly processed seven of the eight animations in acquisition mode and then processed the final animation in description mode. This scenario was repeated for ten runs. The results are shown in table 6. EBLA successfully described 96.67% of the entities with only one unknown entity and no incorrectly named entities.

For the second run, EBLA randomly processed six of the eight animations in acquisition mode and then processed the final two animations in description mode. This scenario was again repeated for ten runs. EBLA successfully described 90% of the 60 entities. The remaining 10% consisted of six unknown entities and no incorrectly named entities. Due to the small size of the

test set, the results were heavily dependent on the order in which the experiences were processed. In spite of this, the results were quite promising and demonstrated the basic abilities of the EBLA Model quite well.

Table 6. Animation Description Results When Describing One of Eight Experiences

Orig. Description	Generated Description	# Correct	# Incorrect	# Unknown
hand touch cube	hand, touch, cube	3.00	0.00	0.00
hand slide cube	hand, slide, cube	3.00	0.00	0.00
hand pickup cube	cube, pickup, hand	3.00	0.00	0.00
hand putdown cube	cube, [unknown], hand	2.00	0.00	1.00
hand putdown ball	ball, putdown, hand	3.00	0.00	0.00
hand pickup ball	ball, pickup, hand	3.00	0.00	0.00
hand touch ball	hand, touch, ball	3.00	0.00	0.00
hand putdown cube	cube, putdown, hand	3.00	0.00	0.00
hand putdown ball	ball, putdown, hand	3.00	0.00	0.00
hand pickup ball	ball, pickup, hand	3.00	0.00	0.00
Totals:		29.00	0.00	1.00

6.4 – Creation of Videos

Based on EBLA’s success with the animation test set, a much larger test set of real videos was created. The videos were filmed using a Canon 3CCD Digital Video Camcorder. A white sheet was hung on a wall and draped over a table to create a neutral background. Two freestanding reading lamps were used in conjunction with two overhead florescent fixtures to supply lighting. Photos of the stage are shown in figure 26.

The camcorder was connected to a PC using a Dazzle Digital Video Creator 80 USB capture card. The videos were captured at a resolution of 320 x 280 using both the JMStudio software included with the Java Media Framework and an open-source video editor called VirtualDub. Several events were filmed at a time in two to three minute blocks using a capture rate of fifteen-frames-per second. These blocks were then separated into individual events of about thirty frames each using VirtualDub and saved using a ten frame-per-second display rate.

Both the original and edited videos were uncompressed to eliminate any side effects that compression artifacts might have had on the vision system.

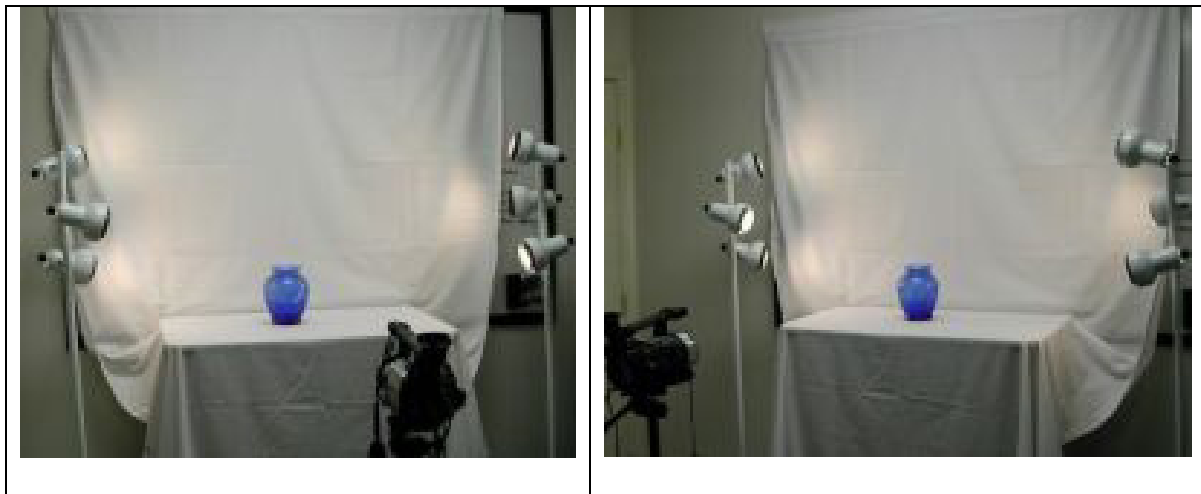


Figure 26. Stage Used to Film EBLA Experiences

All of the videos were shot two to three times from both the left and right side of the stage, each portraying a hand performing some action on an object. The objects included a green ball, a blue glass vase, a dark blue box, a red book, red and orange cups, blue and green colored rings from a toddler toy, small blue and green plastic bowls, and a stuffed Garfield toy. The actions included, *push*, *pull*, *slide*, *touch*, *tipover*, *roll*, *pickup*, *putdown*, *drop*, and *tilt*. Multiple actions were performed on each object, but the actual object-event combinations varied somewhat based on what was feasible for each object. Dropping the glass vase, for example, seemed a bit risky. Sample frames from the videos can be seen in figure 13 in section 4.5, and a full listing of both the videos and the animations in the **experience_data** table can be found in appendix A.

6.5 – Evaluation of Videos

Unfortunately, selecting good segmentation parameters for the real videos was a difficult process. This was due primarily to issues with lighting and shadow. The two freestanding lamps

helped to reduce some shadows, but could not completely eliminate them. Depending on the angle of the arm and the various objects, some of the videos contained glare. This was particularly evident in the videos containing the glass vase, because the reflection of the lamps could be seen. A wide variety of segmentation parameters and speedup options were evaluated for the videos. Selecting an “ideal” set of parameters proved to be difficult because the relationships among the various settings are nonlinear, making the results somewhat unpredictable.

The first segmentation parameter, h_s , represents the spatial resolution and defines a spatial search window for the mean shift computations. Of all the parameters, h_s seems to have the least impact on the results and the greatest impact on the execution time. On a typical frame setting $h_s = 2, 7$ (default), and 13 changes the segmented image only slightly, but changes the computation time from 0.89 to 6.32 to 20.86 seconds respectively using the medium speedup option and an 800 megahertz processor.

The second segmentation parameter, h_r , represents the color or range resolution and in practical terms determines the extent to which colored regions are merged together. A lower color radius leads to oversegmentation where objects are broken into many small colored regions, and a higher color radius leads to undersegmentation where distinct objects are merged together. If h_r is too low, significant objects are split into multiple objects, and if it is too high, objects are merged into the background, effectively disappearing.

Depending on levels of glare and shadow and on object size and texture, a single color resolution can lead to both over and undersegmentation in the same experience. Figure 27 shows the polygon tracings for three frames from a single video shot with the Garfield toy. The frame on the left is correctly segmented, the frame in the middle is undersegmented where the hand has

been merged into the background and essentially disappears, and the frame on the right is oversegmented where the Garfield toy has been split into two objects.

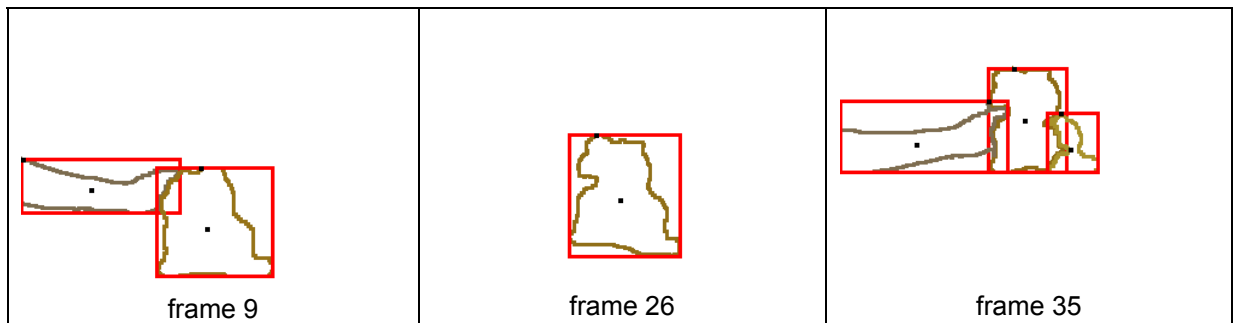


Figure 27. Polygon Traces from a Single Video Demonstrating Normal Segmentation, Undersegmentation, and Oversegmentation

The third and final segmentation parameter, M , determines the minimum region area in pixels. This value is distinct from the parameter in EBLA that establishes the minimum area for an object entity. Smaller values for M , such as twenty, lead to small regions in the middle and on the edge of objects where glare and shadows are detected. Larger values, such as 500, merge shadows and glare into objects, but may also merge enclosed areas of background such as the center of a ring or the space between a hand and the object it is holding. While M and the minimum object entity area in EBLA can differ, using a larger value between 300 and 700 for both generally produces better results. Using too high a value for either parameter results in some of the smaller significant objects being dropped.

The three speedup options: no-speedup, medium-speedup, and high-speedup also affect the segmentation results. The higher the level of speedup, the lower the segmentation quality. The default speedup setting of medium generally produces good results that are not significantly different from the no-speedup setting. Medium-speedup does, however, require a higher color resolution, h_r . Using the high-speedup option is significantly faster, but results in segmented regions with very jagged edges. For a typical frame, using segmentation parameters of $h_s = 7$, h_r

= 6.5, and $M = 20$, the computation times on an 800 megahertz processor are 10.84, 6.28, and 0.45 seconds for the no-speedup, medium-speedup, and high-speedup options respectively.

In order to determine an optimal set of segmentation parameters, the polygon and segmentation images created as part of EBLA's intermediate results were visually inspected. This required loading all of the videos into the **experience_data** table of the EBLA database, and executing both the *FrameGrabber* and *FrameProcessor* classes for each. After trying a wide variety of parameters and speedup options, $h_s = 7$, $h_r = 13.0$, $M = 500$, and medium-speedup were found to provide the most consistent results for all videos.

While using these settings for the entire set of 319 videos, eighty-one dropped a significant object in one or more frames due to undersegmentation, five merged the hand and the object it was acting upon due to undersegmentation, thirty-one split a significant object in one or more frames due to oversegmentation, two shrunk a significant object due to failed polygon trace, three dropped and split significant objects in the same video, and 197 were segmented correctly. Since EBLA treats objects that disappear and then reappear as distinct objects, all of the undersegmented images were manually discarded from the test set. To handle oversegmentation, the *minFrameCount* variable in the *Params* class was set to seven. This allowed EBLA to discard portions of objects that were split for just a few frames. The *minFrameCount* would have been set slightly higher, but there were a few properly segmented videos with only eight frames.

Of the 319 videos, 226 were delivered to EBLA for evaluating lexical acquisition performance. This test set was established by including all experiences that had three entities following the entity extraction phase. Some of the entities were probably detected incorrectly since various combinations of oversegmentation and undersegmentation along with extra objects

caused by shadows could have led to situations where there were three object entities and no relation entities. Such “perceptual noise” just made the acquisition task that much more realistic.

Just as with the animations, the 226 video experiences were delivered to EBLA at random, ten times for each of nineteen different minimum standard deviation (σ_{\min}) values, yielding a total of 190 runs. The value of σ_{\min} used to match attribute values to existing entities was again varied from 5% to 95% in increments of 5%. Figure 28 shows the average acquisition time (in number of experiences) based on the number of experiences processed. A third-order polynomial curve was fit to the data. What this graph indicates is that, for the first few experiences, it took an average of over twenty experiences to resolve all of the entity-lexeme mappings. After about seventy-five experiences had been processed, this average dropped to about five experiences, and after about 150 experiences, the average fell below one.

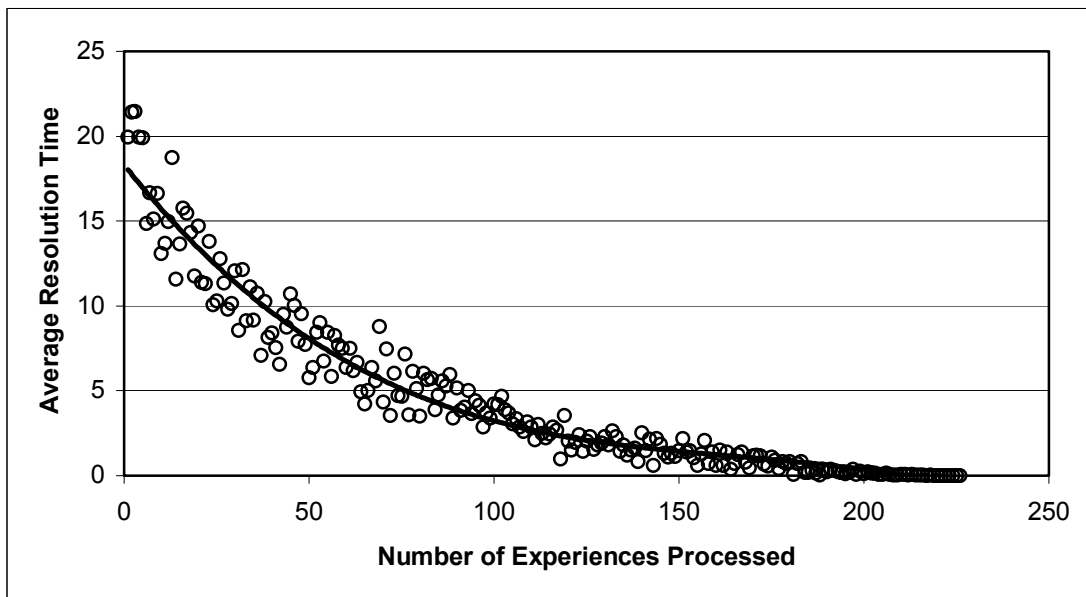


Figure 28. Average Lexical Acquisition Time for Videos

Figure 29 shows the success rates for lexeme mappings for each of the nineteen σ_{\min} values. The results are quite different from those of the animation experiences. For σ_{\min} values of 5% and 10%, the acquisition success was only 76% and 85% respectively. This can be

attributed to the amount of variation in the entities for the videos. A stricter matching criteria results in more unmatched entities. For all of the other σ_{\min} values the acquisition success rate was better than 90% and as high as 95.8% for a σ_{\min} value of 45%. Considering the subjective nature of the videos, EBLA performed quite well.

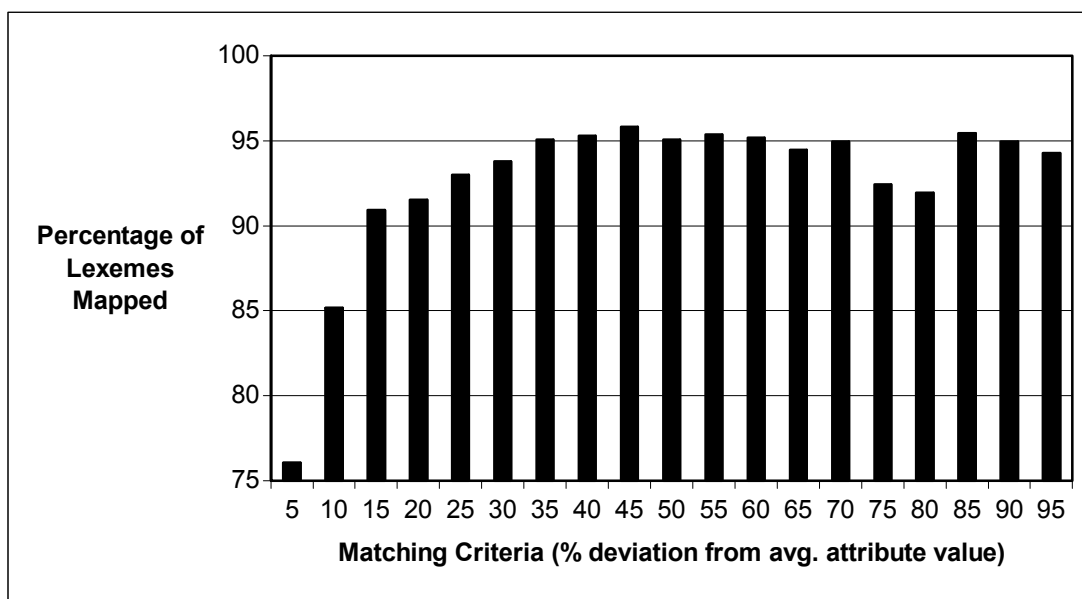


Figure 29. Lexeme Mapping Success Rates for Different Minimum Standard Deviations

The test set used to evaluate EBLA’s ability to generate descriptions for the videos was reduced from 226 experiences to 167 in order to remove all known videos with segmentation issues. It was not particularly useful to see how well EBLA names phantom or split objects. Of the 167 videos chosen, EBLA randomly processed 157 in acquisition mode and then processed ten in description mode. This scenario was run ten times for each of the same nineteen σ_{\min} values used to evaluate acquisition success. The results are shown in table 7 and figure 30.

For the lower values σ_{\min} , there were very few incorrect descriptions, but many entities did not map to a known lexeme. As σ_{\min} was increased, the situation reversed with almost every entity mapping to some lexeme, but many to the wrong lexeme. The most accurate descriptions were produced for a σ_{\min} value of 15% where just over 65% of the entities were described

correctly. These are reasonably good results considering the amount that any given entity varied from video to video. In addition, many entities mapped to multiple lexical items, especially for the higher values of σ_{\min} . To handle this situation, all lexical matches were added to the descriptions (separated by “OR”), and the correct and incorrect mappings totals were incremented on a pro-rata basis. For example, if a bowl object entity mapped to the lexemes “bowl,” “box,” and “ring,” the number of correct mappings would be increased by 1/3 and the number of incorrect mappings would be increased by 2/3. The large number of incorrect mappings for the higher values of σ_{\min} can be primarily attributed to these multiple mappings.

Table 7. Accuracy of Video Descriptions

σ_{\min}	% Correct	% Incorrect	% Unknown
5	50.33	9.00	40.67
10	57.22	14.11	28.67
15	65.33	16.00	18.67
20	56.07	25.27	18.67
25	57.44	27.89	14.67
30	62.94	27.73	9.33
35	59.30	35.03	5.67
40	63.14	30.52	6.33
45	60.95	34.05	5.00
50	50.83	41.17	8.00
55	55.04	40.62	4.33
60	48.39	45.94	5.67
65	46.21	49.46	4.33
70	49.96	45.38	4.67
75	43.63	53.03	3.33
80	44.42	50.91	4.67
85	46.45	50.55	3.00
90	45.04	52.62	2.33
95	39.51	54.49	6.00

A small subset of the descriptions generated by EBLA for the video test set using a σ_{\min} value of 15% are shown in table 8. One of the problems that this table illustrates is the existence of bad mappings in the database. For example, the second description produced by EBLA indicates that the lexemes “tilt” or “hand” could be mapped to the same entity. This is obviously

incorrect as one lexeme describes an object entity and the other describes a relation entity. Unfortunately, these bad mappings had an adverse effect on EBLA’s ability to generate descriptions.

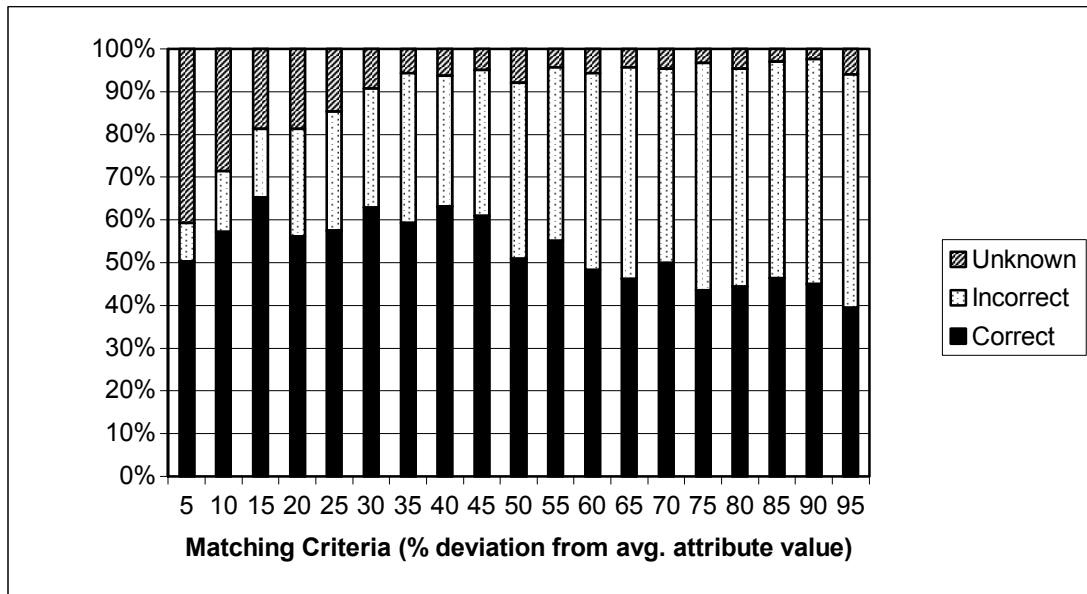


Figure 30. Distribution of Correct, Incorrect, and Unknown Lexemes in Video Descriptions

Table 8. Sample Video Description Results for Ten of 167 Video Experiences

Orig. Description	Generated Description	# Correct	# Incorrect	# Unknown
hand push box	box, hand, push	3.00	0.00	0.00
hand touch ring	tilt OR hand, [unknown],ring	1.50	0.50	1.00
hand putdown bowl	bowl, hand, putdown OR pickup	2.50	0.50	0.00
hand putdown book	putdown OR pickup, hand, book	2.50	0.50	0.00
hand tipover box	box, touch, tipover	2.00	1.00	0.00
hand pickup book	book, hand, putdown OR pickup	2.50	0.50	0.00
hand touch ball	ball, hand, pull OR touch	2.50	0.50	0.00
hand roll ring	[unknown], [unknown], [unknown]	0.00	0.00	3.00
hand pull bowl	ring OR bowl, [unknown],pull OR touch	1.00	1.00	1.00
hand touch box	[unknown],ring OR hand, touch OR hand	1.00	1.00	1.00
hand pickup book	book, hand, pickup OR putdown	2.50	0.50	0.00
hand tilt ring	[unknown],hand, touch OR tipover OR ring	1.33	0.67	1.00
hand pull bowl	ring OR bowl, [unknown],pull OR touch	1.00	1.00	1.00
hand pickup ball	ball, hand, pickup OR putdown	2.50	0.50	0.00

The bad entity-lexeme mappings in the **ebla_data** database resulted from the combination of an improperly segmented video and the lexical resolution phase that

automatically generates entity-lexeme mappings whenever an experience has a single unmapped entity and a single unmapped lexeme. For example, if, in a video of a hand tilting a ring, EBLA created two hand object entities by dropping the hand for a single frame and did not detect the contact between the hand and the ring, it could easily map the lexeme “tilt” to one of the hands. While the results of the segmentation process were visually inspected to filter out poorly segmented videos, the process involved reviewing over 10,000 files, and it is almost a certainty that at least some mistakes were made.

6.6 – Summary

This chapter has detailed the generation of test data for EBLA, the testing procedures used to evaluate EBLA, and the results. The next chapter outlines plans for future work and for the dissemination of EBLA.

CHAPTER 7 – FUTURE WORK AND DISSEMINATION

7.1 – Introduction

The EBLA Model could be extended in many ways to produce additional research results and to increase its practical application. The possible extensions involve everything from creating more complex test sets to incorporating a virtual vision system. This chapter summarizes several of these potential avenues for expansion and discusses how EBLA will be disseminated to the public.

7.2 – More Complex Experiences

The capabilities of EBLA could be further evaluated by testing it with a series of more complex experiences. To date, the only experiences delivered to EBLA have been limited to two objects and a single object-object relation. Limiting EBLA to three entities made debugging and evaluating the model much easier. However, EBLA is theoretically capable of processing single experiences with multiple objects and object-object relations. For example, there is nothing to prevent it from watching two events in the same video, either sequentially or in parallel. EBLA has already demonstrated that it can learn in the presence of extra, unrelated objects such as shadows that are detected as significant objects.

In addition to visual complexity, EBLA's experiences could be made more complex lexically. EBLA has not been evaluated using descriptions with mislabeling, multiple events, or extra lexemes. The animation test set was, however, evaluated using descriptions containing a few articles (e.g. "a hand pickup the ball"). Generally speaking EBLA treated such lexemes as noise, and correctly established entity-lexeme mappings for the remaining words. The results varied quite a bit based on the number of articles and the order in which the small animation set

was evaluated. A larger data set and a more formal evaluation technique are needed before any conclusive results can be obtained.

7.3 – Compound Entities

As mentioned in section 4.6, EBLA has no capacity to process compound entities. Object entities are limited to the single-color segmented regions returned by the vision system, and relation entities are limited to spatial relationships between two object entities. EBLA could conceivably be extended to recognize compound entities by linking object entities that appear to be permanently connected across an entire experience (e.g. a head and torso) and to link relation entities that involve common object entities (e.g. stack or play). One approach would be to add a **compound_entity_data** table to the **ebla_data** database. This table could be linked to the existing **entity_data** table in a one-to-many fashion. Consideration would need to be given to positional relationships among object entities (e.g. head over torso) and sequential relationships among relation entities. To manage the latter, a Petri-net notation such as Bailey's executing schemas (x-schemas) may be appropriate.

7.4 – Model-Merging

Currently EBLA does not have a way to drop attributes from an entity's definition. For example, based on the grayscale attribute, objects that are identical in every aspect except color are often recognized as distinct entities. A model-merging capability similar to the one employed by Bailey could be used to overcome this. A background process could be added to query entities with identical lexical mappings that differ by only one or two attribute values. These entities could then be merged into a single entity without the problematic attribute.

Such a mechanism may also offer a way to acquire additional parts-of-speech. Attributes that are dropped from an entity's description could be used to form new entities. These new

entities could then be added back to the pool of unmapped entities for the original experience. For example, consider two experiences: “hand pickup big box” and “hand pickup small box.” If the *box* entities for the two experiences were merged by dropping the area attribute, the two area attribute values could be used to form new single-attribute entities. EBLA could then conceivably map these new entities to “big” and “small.” It should be noted that before any model-merging feature is incorporated, more attributes should be added to EBLA to allow it to correctly distinguish among entities in the presence of dropped attributes.

7.5 – Dynamic Attributes

As mentioned in section 5.8, the **attribute_list_data** table has a field to hold the name of a calculation class for each attribute. The *EntityExtractor* class could easily be enhanced with the Java *forName()* method to dynamically load attribute calculations at run time without recompiling. Such a feature would allow users to write their own attribute calculation classes and to incorporate them into EBLA with only a simple database edit.

7.6 – Graphical User Interface

Currently EBLA is executed from a command prompt. In addition, changes to the **ebla_data** database for EBLA must be made via the psql command line PostgreSQL client, the Windows-based PGAdmin graphical PostgreSQL client, or the Linux-based PGAccess graphical PostgreSQL client. Ideally, a Java-based graphical user interface (GUI) should be added to EBLA to make it more user-friendly. Construction has been started on a basic GUI for EBLA using Java Swing components, but it is not yet ready for general use.

7.7 – Speech Recognition and Speech Synthesis

There are freely available Java application program interfaces (API's) for speech recognition and speech synthesis. The speech recognition API could be used to allow EBLA to

process experiences containing audio descriptions as well as video. In addition, the speech synthesis API could be used to allow EBLA to generate verbal scene descriptions. While such enhancements are not essential to the basic functionality of EBLA, they would make its acquisition and description capabilities more realistic.

7.8 – Syntax

EBLA may be able to provide a good foundation for studying the acquisition of syntax and its effect on the acquisition of vocabulary. A syntax module could be coded explicitly based on a rule set, implicitly based on a neural-network or other bottom-up learning algorithm, or by using some hybrid technique incorporating both explicit and implicit components. It is likely that the simple grounded entity structure employed by EBLA would have to be extended to handle more abstract and symbolic representations since some words such as articles do not have direct correlations to grounded knowledge.

7.9 – Virtual Vision System/Game Engine

Computer vision remains an open issue in computer science. While the mean shift segmentation based vision system used in EBLA is quite powerful, it is far from achieving humanlike vision. Now that EBLA has demonstrated its ability to acquire some form of language based on actual videos, it might be wise to extend its capabilities in a virtual environment. A 3-D game engine would probably be the best way to accomplish this.

Game engines are capable of modeling extremely complex environments with a high level of detail. EBLA could be programmed to perceive events that are much more complicated than those delivered to its current vision system. Information about size, volume, distance, texture, speed, etc. can be obtained quickly and easily just by querying the game engine. This information would allow for the addition of many new attributes, which, in turn, may make

enhancements such as the model-merging mechanism easier to implement. While not a game engine per se, the Java 3-D API is a 3-D graphics engine that would supply much of the functionality of a full-blown game engine and would be fairly straightforward to incorporate into EBLA.

7.10 – Feedback

Currently EBLA can only operate in an unsupervised mode. There is no mechanism for it to make corrections based on its own performance. To improve upon this, both self-supervision and external supervision features could be added. Self-supervision would allow EBLA to correct itself based on available information. For example, the description generation routines could “cheat,” making adjustments dynamically based on known descriptions. This would equate to an explicit training mode. External supervision would allow EBLA to correct itself based on human feedback. Such feedback could be supplied during both the acquisition and description tasks.

7.11 – Knowledge Base

To extend EBLA beyond perceptually grounded entities, some form of external symbolic storage (ESS) systems could be linked to the system. A knowledge base such as the OpenCyc system (<http://www.opencyc.com>) could be incorporated into EBLA to provide it with access to general knowledge and common sense reasoning. As an example of how EBLA might conceivably use such information, consider how humans incorporate information that is not experienced first hand into their grounded perceptual knowledge: if a person has seen a red apple and a green ball, he/she can simply be told that apples can be green and balls can be red without actually *experiencing* either. More abstract information, lacking a direct tie to grounded

perception, could conceivably be incorporated into EBLA using some sort of analogy mechanism.

7.12 – Emotion, Motivation, and Goals

As a final extension to EBLA, an emotion agent module could be added to supply some form of emotion, motivation, and goals. Currently, EBLA could not be seriously classified as an intelligent system. Even if EBLA could one day be made to fully acquire language, it would not know what to do with it! The addition of simulated emotion and motivation to EBLA might allow it to act in an interesting and meaningful way, providing insight into some of the more subjective areas of cognitive development and language acquisition. The research on emotional agents by Cañamero (Cañamero 1997) and Allen (Allen 2000), for example, is based on cognitive and developmental research and could be very synergistic with EBLA.

7.13 – Dissemination

Upon completion of this research, the source code for EBLA will be made publicly available. One of the fundamental goals in the design of EBLA has been to create a system that could be readily extended by others for future research. In addition to documenting the entire system with JavaDoc, the Concurrent Versions System (CVS) has been employed for the latter half of the project to manage coding revisions. CVS is a popular, open-source system for maintaining both source code and system documentation, and can support distributed projects with many developers.

All of the technologies needed to use and extend EBLA are freely available including Java, PostgreSQL, and CVS. The two programs used to capture and edit the entire test video set for EBLA, VirtualDub and the Java Media Framework, are also available free of charge. The source code for EBLA along with information and resources for its related technologies will be

disseminated via <http://www.greatmindsworking.com>, a web portal focused on research on human language acquisition modeling. Additional resources for EBLA are listed in appendix B.

7.14 – Summary

This chapter has outlined a number of possible extensions to the EBLA software framework and has discussed plans for dissemination of the project. The next chapter contains some final remarks on this research.

CHAPTER 8 – CONCLUSIONS

8.1 – A Step in the Right Direction

So, how well does EBLA satisfy David G. Stork's vision of computer-generated scene analysis discussed in chapter 1? In many senses, it barely scratches the surface. EBLA would most certainly choke on video footage of Grand Central Station! The goal for EBLA, however, was not full-blown scene analysis, but to provide computers with *basic* scene description capabilities and a more humanlike capacity for language. EBLA has succeeded on both of these fronts, and has hopefully provided a foundation on which even better systems can be constructed.

While there is no claim that EBLA learns language using the same mechanisms that humans do, EBLA does take an experiential approach to learn language from scratch. EBLA forms simplistic representations for an event based on both the appearance and interactions of objects. These are roughly based on the event representations formed by infants. Over multiple experiences, EBLA abstracts and generalizes its representations, and it begins to correlate them with protolanguage. In a similar fashion, the first words of children take the form of protolanguage, and emerge as children begin to generalize their event representations into conceptual categories. Giving a computer the ability to incorporate language into abstracted representations of grounded perceptual experience establishes a common frame of reference for human and computer. This may one day help man and machine to communicate in more humanlike terms.

EBLA has successfully detected both objects and object-object relations for a set of simple animations as well as for a larger set of videos. The latter is a particularly significant accomplishment for the system considering the amount that any given object varied across the

set of videos. EBLA performed very well on the entity-lexeme mapping task for both the animations and the videos, achieving success rates as high as 100% and 95.8% respectively.

EBLA was also able to generate descriptions for the animations and videos with average accuracies as high as 96.7% and 65.3%. The 65.3% is still quite good when compared to the approximately 15% average success rate obtained by generating three word descriptions at random from the pool of nineteen lexemes processed by EBLA. The lower description success rate for the videos is attributed to a combination of the variance in the attribute values defining the entities, and the fact that some entities map to multiple lexemes. This is more of a shortcoming of the computer vision system than of the lexical acquisition system.

While the segmentation-based vision system for EBLA worked extremely well on a video-by-video basis in that almost every video could be segmented properly by tuning the segmentation parameters, no set of parameters was discovered that worked well for the entire set of 319 videos. Although using a higher quality test set created with studio quality lighting and background would probably result in better segmentation and a wider range of acceptable segmentation parameters, the videos would be less realistic and more difficult for other experimenters to replicate.

8.2 – How Does EBLA Compare?

It is difficult to make quantitative comparisons between EBLA and related computational models of language acquisition because of the differences in design goals. For example, comparing the acquisition rates between EBLA and Siskind's word-to-meaning mapping system is a pointless exercise. Siskind's system is based on *symbolic* representations, while EBLA is based on *grounded* representations. Siskind's system acquires hundreds of words representing multiple parts of speech by processing thousands of symbolic representation-utterance pairings.

EBLA acquires less than twenty words representing protolanguage nouns and verbs by processing a few hundred video-description pairings. A better way to evaluate EBLA is on a qualitative basis. Table 9 is an extension of table 1 in section 3.8 and compares EBLA with related models on a feature-by-feature basis.

Table 9. Comparison of EBLA with Other Computational Models

Model / Feature	Bottom-Up	Parts-of-Speech	Syntax	Perception	Speech Processing	Socially Mediated	Source Available
Bailey (VerbLearn)	requires explicit training	action verbs	no	virtual proprioception	text	no	yes
Pangburn (EBLA)	yes	object nouns and action verbs	no	vision (segmentation-based)	text	no	yes
Roy (CELL)	yes	shape and color words	no	vision (histogram-based)	audio	no	no
Roy (DESCRIBER)	requires explicit training	spatial description words	yes	virtual vision	text in / audio out	no	no
Siskind (HOWARD & LEONARD)	requires explicit training	event labels	no	vision (segmentation-based)	text	no	yes
Siskind (word-to-meaning mappings)	yes	multiple	no	virtual "meaning" symbols	text	no	yes
Steels & Kaplan (AIBO)	utilizes base lexicon	object nouns	no	vision (histogram-based)	audio	yes	no

Comparing perceptual systems, the HOWARD and LEONARD event recognition systems of Siskind, the CELL system of Roy, the AIBO™ software of Steels and Kaplan, and EBLA all integrate computer vision systems capable of processing real video. HOWARD, LEONARD, and EBLA use image segmentation as the basis for their vision systems, while the others use color histograms. The DESCRIBER system of Roy uses a virtual vision system to perceive its environment, although work is in progress to incorporate real video. Bailey's VerbLearn system uses virtual proprioception rather than vision as the basis for its perception.

Siskind's system for learning word-to-meaning mappings does not incorporate perception, and instead grounds language to symbolic meanings.

Comparing acquisition systems, only EBLA can acquire a grounded lexicon for both objects (i.e. nouns) and object-object relations (i.e. verbs). Siskind's HOWARD and LEONARD systems acquire single words to classify visual events, but do not address object naming. His word-to-meaning mapping software acquires a wide variety of speech for symbolic representations using cross-situational learning. Roy's CELL system correlates audio signals to visual data to perform lexical segmentation and acquisition of color and shape words. His DESCRIBER system acquires noun phrases and spatial clauses based on transcribed speech for computer-generated images of rectangles. Steels and Kaplan's AIBO software acquires simple object names based on visual data and audio feedback. Bailey's VerbLearn software uses linking feature structures (f-structs) to correlate verbs to executing-schema (x-schema) representations of proprioceptively perceived actions. Of all of these systems, only Roy's DESCRIBER system and Siskind's word-to-meaning mapping system attempt to incorporate syntax, although Bailey does discuss VerbLearn extensions for the acquisition of verb affixes, auxiliaries, and particles.

All of the systems except the AIBO software are primarily bottom-up in that they are not explicitly pre-programmed with perceptual or lexical information. The AIBO software is supplied with a series of basic commands, some question structures (e.g. "What is it?"), and some feedback words (e.g. "yes" and "no"). The learning techniques employed vary from neural networks to probabilistic methods to inference. Some operate in an online fashion in that there is no explicit training set, while others are trained and evaluated using separate data sets. EBLA

does not require explicit training in order to generate entity-lexeme mappings, but it must have a certain number of mappings established before descriptions can be generated.

As a final comparison, only the CELL system and the AIBO software are capable of receiving lexical information in the form of an audio signal. These systems, as well as the DESCRIBER system, are also capable of performing speech synthesis to generate descriptions. All of the other systems, including EBLA, are text-based for both input and output.

8.3 – Applications

The applications of a full-blown scene description/natural language system would be numerous. They include robotics, security/surveillance, language translation, and even aid for the perceptually impaired, to name a few. Such a system, however, is still likely many years away. In the meantime, a limited system such as EBLA may still have several practical applications. Although the vision system employed for EBLA is not perfect, its ability to recognize basic objects and object-object relations could be useful for tasks such as robot vision and path planning. EBLA's language acquisition system may have application in such fields as game artificial intelligence (AI) where it could be used to build more intelligent "bots" capable of basic communication with players. The most significant and immediate application for EBLA, however, will probably be additional research.

8.4 – Summary

This chapter has evaluated the success of the EBLA software system and discussed possible applications. The principal contributions of this research have been:

1. EBLA is the first known system to acquire both nouns *and* verbs based on grounded perception.
2. EBLA operates in a bottom-up fashion without an explicit training phase.

3. EBLA is based on both existing computational and developmental research.
4. EBLA can perform basic scene analysis.
5. EBLA provides an open framework for additional research.

While computational models of human language acquisition are still in their infancy, the early successes of EBLA and related models will hopefully help to draw others into this promising area of research.

REFERENCES

- Allen, Stephen Richard. 2000. Concern processing in autonomous agents. Ph.D. diss., University of Birmingham. Available <http://www.dfki.de/~allen/Thesis/index.html>.
- Bailey, David Robert. 1997. When push comes to shove: A computational model of the role of motor control in the acquisition of action verbs. Ph.D. diss., University of California, Berkeley. Available <http://www.icsi.berkeley.edu/~dbailey/diss.pdf>.
- Bailey, David Robert, Jerome Feldman, Srini Narayanan, and George Lakoff. 1997. Modeling embodied lexical development. In *Proceedings of the nineteenth annual conference of the Cognitive Science Society held in Stanford, CA, August 7-10, 1997*, edited by Michael G. Shafto and Pat Langley. Mahwah, NJ: Lawrence Erlbaum Associates, Inc. Available <http://www.icsi.berkeley.edu/~dbailey/cogsci97.pdf>.
- Bailey, David Robert, Nancy Chang, Jerome Feldman, and Srini Narayanan. 1998. Extending embodied lexical development. In *Proceedings of the twentieth annual conference of the Cognitive Science Society held in Madison, WI, August 1-4, 1998*, edited by Morton Ann Gernsbacher and Sharon J. Derry. Mahwah, NJ: Lawrence Erlbaum Associates, Inc. Available <http://www.icsi.berkeley.edu/~dbailey/cogsci98.pdf>.
- Brent, Michael R. and Jeffrey Mark Siskind. 2001. The role of exposure to isolated words in early vocabulary development. *Cognition* 81, no. 2 (September): 33-44. Available <ftp://ftp.ecn.purdue.edu/qobi/cognition2001.pdf>.
- Calvin, William H. and Derek Bickerton. 2001. *Lingua ex machina: Reconciling Darwin and Chomsky with the human brain*. Cambridge, MA: MIT Press, 2000. Reprint, MIT Press (page references are to the reprint edition).
- Cañamero, Dolores. 1997. Modeling motivations and emotions as a basis for intelligent behavior. In *Proceedings of the first international conference on autonomous agents (Agents '97) held in Marina Del Rey, CA, February 5-8, 1997*, edited by W. Lewis Johnson, 148-155. New York, NY: The ACM Press. Available <http://homepages.feis.herts.ac.uk/~comqlc/papers/aa97-online.ps.gz>.
- Cederqvist, Per, et. al. 2001. *Version management with CVS for CVS 1.11.1p1. [electronic document]*. Available <http://www.cvshome.org/docs/manual/>.
- Christoudias, Christopher M., Bogdan Georgescu, and Peter Meer. 2002. Synergism in low level vision. In *Proceedings of the sixteenth international conference on pattern recognition (ICPR 2002) held in Quebec City, Canada, August 11-15, 2002*. New York, NY: IEEE Press. Available <http://www.caip.rutgers.edu/riul/research/papers/pdf/segedge.pdf>.

- Comaniciu, Dorin. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE transactions on pattern analysis and machine intelligence* 24, no. 5 (May): 603-619. Available <http://www.caip.rutgers.edu/~comanici/Papers/MsRobustApproach.pdf>.
- Comaniciu, Dorin and Peter Meer. 1997. Robust analysis of feature spaces: Color image segmentation. In *Proceedings of the 1997 conference on computer vision and pattern recognition (CVPR '97) held in San Juan, Puerto Rico, June 17-19, 1997*, 750-755. New York, NY: IEEE Press. Available <http://www.caip.rutgers.edu/riul/research/papers/pdf/feature.pdf>.
- _____. 1999. Mean shift analysis and applications. In *Seventh IEEE international conference on computer vision (ICCV '99) held in Kerkyra, Corfu, Greece, September 20-25, 1999*, 1197-1203. New York, NY: IEEE Press. Available <http://www.caip.rutgers.edu/~comanici/Papers/MsAnalysis.pdf>.
- de Jong, Edwin D. 2000. Autonomous formation of concepts and communication. Ph.D. diss., Vrije Universiteit Brussel. Available <http://www.cs.brandeis.edu/~edwin/thesis/>.
- Elman, Jeffrey L., Elizabeth A. Bates, Mark H. Johnson, Annette Karmiloff-Smith, Domenico Parisi, and Kim Plunkett. 1999. *Rethinking innateness: A connectionist perspective on development*. Cambridge, MA: MIT Press, 1996. Reprint, MIT Press (page references are to the reprint edition).
- Gillis, Justin. 2002. Gene mutations linked to language development. In, *washingtonpost.com [online journal]*. [cited 2002-08-15]. Available <http://www.washingtonpost.com/ac2/wp-dyn/A17863-2002Aug14>.
- Grand, Steve. 2000. *Creation: Life and how to make it*. London, England: Weidenfeld & Nicolson.
- Hardy, Vincent J. 2000. *Java: 2D API graphics*. Palo Alto, CA: Sun Microsystems Press.
- Holland, John H. 1996. *Hidden order: How adaptation builds complexity*. Reading, MA: Addison-Wesley, 1995. Reprint, Addison Wesley (page references are to the reprint edition).
- Jurafsky, Daniel and James H. Martin. 2000. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, NJ: Prentice Hall.
- Kauffman, Stuart. 1993. *The origins of order: Self-organization and selection in evolution*. New York, NY: Oxford University Press.
- Kauffman, Stuart. 1995. *At home in the universe: The search for the laws of self-organization and complexity*. New York, NY: Oxford University Press.

- Lakoff, George. 1990. *Women, fire, and dangerous things: What categories reveal about the mind*. Chicago, IL: The University of Chicago Press, 1987. Reprint, The University of Chicago Press (page references are to the reprint edition).
- Locke, John L. 1993. *The child's path to spoken language*. Cambridge, MA: Harvard University Press.
- Lyon, Douglas A. 1999. *Image processing in Java*. Upper Saddle River, NJ: Prentice Hall PTR.
- Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. Cambridge, MA: MIT Press.
- MacWhinney, Brian. 1998. Models of the emergence of language. *Annual review of psychology* 39, no. 1: 199-227. Available <http://psyling.psy.cmu.edu/papers/annual.pdf>.
- Miller, George A. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review* 63: 81-97. Available <http://www.well.com/user/smalin/miller.html>.
- Momjian, Bruce. 2000. *PostgreSQL: Introduction and concepts*. Reading, MA: Addison-Wesley. Available <http://www.ca.postgresql.org/docs/awbook.html>.
- Naughton, Patrick and Herbert Schildt. 1999. *Java: The complete reference*. 3d ed. Berkeley, CA: Osborne/McGraw-Hill.
- Nelson, Katherine. 1998. *Language in cognitive development: The emergence of the mediated mind*. Cambridge, UK: Cambridge University Press, 1996. Reprint, Cambridge University Press (page references are to the reprint edition).
- Norris, Janet A. and Paul R. Hoffman. 2002. Language development and late talkers: A connectionist perspective. In *Connectionist approaches to clinical problems in speech and language: Therapeutic and scientific applications*, ed. Raymond G. Daniloff, 1-109. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.
- Pinker, Steven. 2000. *The language instinct: How the mind creates language*. New York, NY: William Morrow and Company, 1994. Reprint, Perennial Classics (page references are to the reprint edition).
- Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1997. *Numerical recipes in C: The art of scientific computing*. 2d ed. Cambridge, UK: Cambridge University Press, 1992. Reprint, Cambridge University Press (page references are to the reprint edition).

- Roy, Deb Kumar. 1999. Learning words from sights and sounds: A computational model. Ph.D. diss., Massachusetts Institute of Technology. Available http://web.media.mit.edu/~dkroy/papers/pdf/phd_thesis.pdf.
- _____. 2000. A computational model of word learning from multimodal sensory input. In *Proceedings of third international conference of cognitive modeling (ICCM 2000), held in Groningen, Netherlands, March 23-25, 2000*. Veenendaal, Netherlands: Universal Press. Available <http://web.media.mit.edu/~dkroy/papers/pdf/iccm2000.pdf>.
- _____. 2000. Learning visually grounded words and syntax of natural spoken language. In *Evolution of communication journal* 4, no. 1 (April): 33-57. Available <http://web.media.mit.edu/~dkroy/papers/pdf/ec.pdf>.
- Siskind, Jeffrey Mark. 1992. Naïve physics, event perception, lexical semantics, and language acquisition. Ph.D. diss., Massachusetts Institute of Technology. Available <ftp://ftp.ecn.purdue.edu/qobi/phd.ps.Z>.
- _____. 1997. A computational study of cross-situational techniques for learning word-to-meaning mappings. In *Computational approaches to language acquisition*, ed. Michael Brent, 39-91. Amsterdam, Netherlands: Elsevier Science Publishers, 1996. Reprint, MIT Press (page references are to the reprint edition). Available <ftp://ftp.ecn.purdue.edu/qobi/cognition96.ps.Z>.
- _____. 1999. *Visual event perception*. Princeton, NJ: NEC Research Institute, Inc. Technical Report 99-033. Available <ftp://ftp.ecn.purdue.edu/qobi/nara98.pdf>.
- _____. 2000. Visual event classification via force dynamics. In *Proceedings of the seventeenth national conference on artificial intelligence (AAAI 2000) held in Austin, TX, July 30-August 3, 2000*, 149-155. Menlo Park, CA: AAAI Press. Available <ftp://ftp.ecn.purdue.edu/qobi/aaai2000.pdf>.
- _____. 2001. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of artificial intelligence research* 15 (August): 31-90. Available <ftp://ftp.ecn.purdue.edu/qobi/jair2001.pdf>.
- Siskind, Jeffrey Mark and Quaid Morris. 1996. A maximum-likelihood approach to visual event classification. In *Proceedings of the fourth European conference on computer vision (ECCV '96), held in Cambridge, UK, April 15-18, 1996*, Vol. 2, 347-360. New York, NY: Springer-Verlag. Available <ftp://ftp.ecn.purdue.edu/qobi/eccv96a.ps.Z>.
- Smith, Emily. 1999. The performance of prekindergarten children on representational tasks across levels of displacement. Ph.D. diss., Louisiana State University.
- Steels, Luc. 1997. The synthetic modeling of language origins. In *Evolution of communication journal* 1, no. 1 (September): 1-34. Available <http://arti.vub.ac.be/steels/coe.ps>.

- Steels, Luc and Frederic Kaplan. 2000. AIBO's first words: The social learning of language and meaning. In *Evolution of communication journal* 4, no. 1 (April): 3-32. Available <http://arti.vub.ac.be/steels/aibo.ps>.
- Stork, David G. 2000. The best-informed dream. In, *HAL's legacy: 2001's computer as dream and reality*, ed. David G. Stork, 1-12. Cambridge, MA: MIT Press, 1997. Reprint, MIT Press (page references are to the reprint edition).
- Sun Microsystems. 1999. *Java Media Framework API guide [electronic document]*. Mountain View, CA: Sun Microsystems. Available <http://java.sun.com/products/java-media/jmf/>.
- Waldrop, M. Mitchell. 1993. *Complexity: The emerging science at the edge of order and chaos*. New York, NY: Touchstone, 1992. Reprint, Touchstone (page references are to the reprint edition).
- Webster's Ninth New Collegiate Dictionary. 1989. s.v. "apple."
- Weiss, Mark Allen. 1999. *Data structures & algorithm analysis in Java*. Reading, MA: Addison-Wesley.
- Whitfield, John. 2001. Language gene found. In, *Nature science update [electronic journal]*. [cited 2002-08-15]. Available http://www.nature.com/nsu/nsu_pf/011004/011004-16.html.

APPENDIX A – LISTING OF EXPERIENCES PROCESSED BY EBLA

experience_id	source_movie	language_text	notes
1	ball_up1.avi	hand pickup ball	animation
2	ball_down1.avi	hand putdown ball	animation
3	ball_slide1.avi	hand slide ball	animation
4	ball_touch1.avi	hand touch ball	animation
5	cube_up1.avi	hand pickup cube	animation
6	cube_down1.avi	hand putdown cube	animation
7	cube_slide1.avi	hand slide cube	animation
8	cube_touch1.avi	hand touch cube	animation
9	ball_20020619_down1L.avi	hand putdown ball	
10	ball_20020619_down1R.avi	hand putdown ball	
11	ball_20020619_down2R.avi	hand putdown ball	
12	ball_20020619_down3R.avi	hand putdown ball	
13	ball_20020619_touch1L.avi	hand touch ball	
14	ball_20020619_touch1R.avi	hand touch ball	
15	ball_20020619_touch2L.avi	hand touch ball	
16	ball_20020619_touch3L.avi	hand touch ball	
17	ball_20020619_up1L.avi	hand pickup ball	
18	ball_20020619_up1R.avi	hand pickup ball	
19	ball_20020619_up2L.avi	hand pickup ball	
20	ball_20020619_up3L.avi	hand pickup ball	
21	ball_20020809_down1R.avi	hand putdown ball	
22	ball_20020809_down2R.avi	hand putdown ball	
23	ball_20020809_drop1L.avi	hand drop ball	
24	ball_20020809_drop1R.avi	hand drop ball	
25	ball_20020809_drop2L.avi	hand drop ball	
26	ball_20020809_drop2R.avi	hand drop ball	
27	ball_20020809_roll1L.avi	hand roll ball	
28	ball_20020809_roll1R.avi	hand roll ball	
29	ball_20020809_roll2L.avi	hand roll ball	
30	ball_20020809_roll2R.avi	hand roll ball	
31	ball_20020809_touch1L.avi	hand touch ball	
32	ball_20020809_touch1R.avi	hand touch ball	
33	ball_20020809_touch2L.avi	hand touch ball	
34	ball_20020809_touch2R.avi	hand touch ball	
35	ball_20020809_up1L.avi	hand pickup ball	
36	ball_20020809_up1R.avi	hand pickup ball	
37	ball_20020809_up2L.avi	hand pickup ball	
38	ball_20020809_up2R.avi	hand pickup ball	
39	bluebowl_20020809_down1L.avi	hand putdown bowl	drop
40	bluebowl_20020809_down1R.avi	hand putdown bowl	
41	bluebowl_20020809_down2R.avi	hand putdown bowl	
42	bluebowl_20020809_drop1L.avi	hand drop bowl	
43	bluebowl_20020809_drop1R.avi	hand drop bowl	drop

44	bluebowl_20020809_drop2L.avi	hand drop bowl	
45	bluebowl_20020809_drop2R.avi	hand drop bowl	drop
46	bluebowl_20020809_pull1L.avi	hand pull bowl	
47	bluebowl_20020809_pull1R.avi	hand pull bowl	
48	bluebowl_20020809_pull2L.avi	hand pull bowl	
49	bluebowl_20020809_pull2R.avi	hand pull bowl	
50	bluebowl_20020809_push1L.avi	hand push bowl	
51	bluebowl_20020809_push1R.avi	hand push bowl	
52	bluebowl_20020809_push2L.avi	hand push bowl	
53	bluebowl_20020809_push2R.avi	hand push bowl	
54	bluebowl_20020809_touch1L.avi	hand touch bowl	
55	bluebowl_20020809_touch1R.avi	hand touch bowl	
56	bluebowl_20020809_touch2L.avi	hand touch bowl	
57	bluebowl_20020809_touch2R.avi	hand touch bowl	
58	bluebowl_20020809_up1L.avi	hand pickup bowl	drop
59	bluebowl_20020809_up1R.avi	hand pickup bowl	
60	bluebowl_20020809_up2L.avi	hand pickup bowl	drop
61	bluebowl_20020809_up2R.avi	hand pickup bowl	
62	bluering_20020619_down1L.avi	hand putdown ring	
63	bluering_20020619_down1R.avi	hand putdown ring	
64	bluering_20020619_down2L.avi	hand putdown ring	
65	bluering_20020619_down2R.avi	hand putdown ring	
66	bluering_20020619_drop1R.avi	hand drop ring	
67	bluering_20020619_drop2R.avi	hand drop ring	
68	bluering_20020619_drop3R.avi	hand drop ring	
69	bluering_20020619_drop4R.avi	hand drop ring	
70	bluering_20020619_pull1L.avi	hand pull ring	
71	bluering_20020619_pull1R.avi	hand pull ring	
72	bluering_20020619_slide1L.avi	hand slide ring	
73	bluering_20020619_slide1R.avi	hand slide ring	
74	bluering_20020619_tilt1L.avi	hand tilt ring	
75	bluering_20020619_tilt1R.avi	hand tilt ring	
76	bluering_20020619_tilt2L.avi	hand tilt ring	
77	bluering_20020619_tilt2R.avi	hand tilt ring	
78	bluering_20020619_tilt3R.avi	hand tilt ring	
79	bluering_20020619_tipover1L.avi	hand tipover ring	
80	bluering_20020619_tipover1R.avi	hand tipover ring	
81	bluering_20020619_tipover2L.avi	hand tipover ring	
82	bluering_20020619_tipover2R.avi	hand tipover ring	
83	bluering_20020619_touch1L.avi	hand touch ring	
84	bluering_20020619_touch1R.avi	hand touch ring	
85	bluering_20020619_touch2L.avi	hand touch ring	
86	bluering_20020619_touch3L.avi	hand touch ring	
87	bluering_20020619_up1L.avi	hand pickup ring	
88	bluering_20020619_up1R.avi	hand pickup ring	
89	bluering_20020619_up2L.avi	hand pickup ring	

90	bluering_20020619_up3L.avi	hand pickup ring	
91	bluering_20020809_down1L.avi	hand putdown ring	drop
92	bluering_20020809_down1R.avi	hand putdown ring	drop
93	bluering_20020809_down2L.avi	hand putdown ring	drop
94	bluering_20020809_down2R.avi	hand putdown ring	drop
95	bluering_20020809_drop1L.avi	hand drop ring	drop
96	bluering_20020809_drop1R.avi	hand drop ring	drop
97	bluering_20020809_drop2L.avi	hand drop ring	drop
98	bluering_20020809_drop2R.avi	hand drop ring	drop
99	bluering_20020809_roll1L.avi	hand roll ring	
100	bluering_20020809_roll1R.avi	hand roll ring	drop
101	bluering_20020809_roll2L.avi	hand roll ring	
102	bluering_20020809_roll2R.avi	hand roll ring	drop
103	bluering_20020809_rollchaseL.avi	hand roll ring	
104	bluering_20020809_touch1L.avi	hand touch ring	drop
105	bluering_20020809_touch1R.avi	hand touch ring	drop
106	bluering_20020809_touch2L.avi	hand touch ring	drop
107	bluering_20020809_touch2R.avi	hand touch ring	drop
108	bluering_20020809_up1L.avi	hand pickup ring	drop
109	bluering_20020809_up1R.avi	hand pickup ring	drop
110	bluering_20020809_up2L.avi	hand pickup ring	drop
111	bluering_20020809_up2R.avi	hand pickup ring	drop
112	book_20020809_down1L.avi	hand putdown book	
113	book_20020809_down1R.avi	hand putdown book	
114	book_20020809_down2L.avi	hand putdown book	
115	book_20020809_down2R.avi	hand putdown book	
116	book_20020809_pull1L.avi	hand pull book	split
117	book_20020809_pull1R.avi	hand pull book	split
118	book_20020809_pull2L.avi	hand pull book	split
119	book_20020809_pull2R.avi	hand pull book	split
120	book_20020809_slide1L.avi	hand slide book	split
121	book_20020809_slide1R.avi	hand slide book	split
122	book_20020809_slide2L.avi	hand slide book	split
123	book_20020809_slide2R.avi	hand slide book	split
124	book_20020809_tilt1L.avi	hand tipover book	split
125	book_20020809_tilt1R.avi	hand tipover book	split
126	book_20020809_tilt2L.avi	hand tipover book	split
127	book_20020809_tilt2R.avi	hand tipover book	split
128	book_20020809_touch1L.avi	hand touch book	
129	book_20020809_touch1R.avi	hand touch book	split
130	book_20020809_touch2L.avi	hand touch book	
131	book_20020809_touch2R.avi	hand touch book	
132	book_20020809_up1L.avi	hand pickup book	
133	book_20020809_up1R.avi	hand pickup book	
134	book_20020809_up2L.avi	hand pickup book	
135	book_20020809_up2R.avi	hand pickup book	

136	box_20020619_down1L.avi	hand putdown box	
137	box_20020619_down1R.avi	hand putdown box	merge
138	box_20020619_down2L.avi	hand putdown box	merge
139	box_20020619_down2R.avi	hand putdown box	merge
140	box_20020619_pull1L.avi	hand pull box	
141	box_20020619_pull1R.avi	hand pull box	drop
142	box_20020619_slide1L.avi	hand slide box	
143	box_20020619_slide1R.avi	hand slide box	drop
144	box_20020619_tipover1L.avi	hand tipover box	
145	box_20020619_tipover1R.avi	hand tipover box	
146	box_20020619_tipover2L.avi	hand tipover box	
147	box_20020619_tipover2R.avi	hand tipover box	
148	box_20020619_touch1L.avi	hand touch box	
149	box_20020619_touch1R.avi	hand touch box	
150	box_20020619_touch2L.avi	hand touch box	
151	box_20020619_touch2R.avi	hand touch box	
152	box_20020619_up1L.avi	hand pickup box	
153	box_20020619_up1R.avi	hand pickup box	drop
154	box_20020619_up2L.avi	hand pickup box	merge
155	box_20020619_up3L.avi	hand pickup box	merge
156	box_20020809_down1L.avi	hand putdown box	
157	box_20020809_down1R.avi	hand putdown box	
158	box_20020809_down2L.avi	hand putdown box	
159	box_20020809_down2R.avi	hand putdown box	
160	box_20020809_drop1L.avi	hand drop box	
161	box_20020809_drop1R.avi	hand drop box	
162	box_20020809_drop2L.avi	hand drop box	
163	box_20020809_drop2R.avi	hand drop box	
164	box_20020809_push1L.avi	hand push box	
165	box_20020809_push1R.avi	hand push box	
166	box_20020809_push2L.avi	hand push box	
167	box_20020809_push2R.avi	hand push box	
168	box_20020809_touch1L.avi	hand touch box	
169	box_20020809_touch1R.avi	hand touch box	
170	box_20020809_touch2L.avi	hand touch box	
171	box_20020809_touch2R.avi	hand touch box	
172	box_20020809_up1L.avi	hand pickup box	
173	box_20020809_up1R.avi	hand pickup box	
174	box_20020809_up2L.avi	hand pickup box	
175	box_20020809_up2R.avi	hand pickup box	
176	cup_20020619_down1L.avi	hand putdown cup	
177	cup_20020619_down1R.avi	hand putdown cup	drop
178	cup_20020619_down2L.avi	hand putdown cup	
179	cup_20020619_down2R.avi	hand putdown cup	drop
180	cup_20020619_pull1L.avi	hand pull cup	
181	cup_20020619_pull1R.avi	hand pull cup	drop

182	cup_20020619_slide1L.avi	hand slide cup	drop
183	cup_20020619_slide1R.avi	hand slide cup	drop
184	cup_20020619_slide2L.avi	hand slide cup	
185	cup_20020619_tipover1L.avi	hand tipover cup	
186	cup_20020619_tipover1R.avi	hand tipover cup	
187	cup_20020619_tipover2L.avi	hand tipover cup	
188	cup_20020619_tipover2R.avi	hand tipover cup	
189	cup_20020619_touch1L.avi	hand touch cup	
190	cup_20020619_touch1R.avi	hand touch cup	
191	cup_20020619_touch2L.avi	hand touch cup	
192	cup_20020619_touch2R.avi	hand touch cup	
193	cup_20020619_up1L.avi	hand pickup cup	
194	cup_20020619_up1R.avi	hand pickup cup	
195	cup_20020619_up2L.avi	hand pickup cup	
196	cup_20020619_up2R.avi	hand pickup cup	drop
197	garfield_20020809_up1L.avi	hand pickup garfield	split
198	garfield_20020809_down1L.avi	hand putdown garfield	split
199	garfield_20020809_down1R.avi	hand putdown garfield	split
200	garfield_20020809_down2L.avi	hand putdown garfield	drop and split
201	garfield_20020809_down2R.avi	hand putdown garfield	split
202	garfield_20020809_drop1L.avi	hand drop garfield	split
203	garfield_20020809_drop1R.avi	hand drop garfield	split
204	garfield_20020809_drop2L.avi	hand drop garfield	split
205	garfield_20020809_drop2R.avi	hand drop garfield	split
206	garfield_20020809_touch1L.avi	hand touch garfield	split
207	garfield_20020809_touch1R.avi	hand touch garfield	drop and split
208	garfield_20020809_touch2L.avi	hand touch garfield	split
209	garfield_20020809_touch2R.avi	hand touch garfield	split
210	garfield_20020809_up1R.avi	hand pickup garfield	split
211	garfield_20020809_up2L.avi	hand pickup garfield	drop and split
212	garfield_20020809_up2R.avi	hand pickup garfield	split
213	greenbowl_20020809_down1L.avi	hand putdown bowl	drop
214	greenbowl_20020809_down1R.avi	hand putdown bowl	
215	greenbowl_20020809_down2L.avi	hand putdown bowl	drop
216	greenbowl_20020809_down2R.avi	hand putdown bowl	
217	greenbowl_20020809_drop1L.avi	hand drop bowl	
218	greenbowl_20020809_drop1R.avi	hand drop bowl	
219	greenbowl_20020809_drop2L.avi	hand drop bowl	
220	greenbowl_20020809_drop2R.avi	hand drop bowl	
221	greenbowl_20020809_pull1L.avi	hand pull bowl	
222	greenbowl_20020809_pull1R.avi	hand pull bowl	
223	greenbowl_20020809_pull2L.avi	hand pull bowl	
224	greenbowl_20020809_pull2R.avi	hand pull bowl	
225	greenbowl_20020809_push1L.avi	hand push bowl	
226	greenbowl_20020809_push1R.avi	hand push bowl	drop
227	greenbowl_20020809_push2L.avi	hand push bowl	drop

228	greenbowl_20020809_push2R.avi	hand push bowl	drop
229	greenbowl_20020809_touch1L.avi	hand touch ring	
230	greenbowl_20020809_touch1R.avi	hand touch ring	
231	greenbowl_20020809_touch2L.avi	hand touch ring	
232	greenbowl_20020809_touch2R.avi	hand touch ring	
233	greenbowl_20020809_up1L.avi	hand pickup bowl	drop
234	greenbowl_20020809_up1R.avi	hand pickup bowl	
235	greenring_20020619_down1L.avi	hand putdown ring	
236	greenring_20020619_down1R.avi	hand putdown ring	drop
237	greenring_20020619_down2L.avi	hand putdown ring	
238	greenring_20020619_down2R.avi	hand putdown ring	drop
239	greenring_20020619_drop1R.avi	hand drop ring	drop
240	greenring_20020619_drop2R.avi	hand drop ring	drop
241	greenring_20020619_drop3R.avi	hand drop ring	drop
242	greenring_20020619_drop4R.avi	hand drop ring	
243	greenring_20020619_pull1L.avi	hand pull ring	
244	greenring_20020619_pull1R.avi	hand pull ring	drop
245	greenring_20020619_slide1L.avi	hand slide ring	
246	greenring_20020619_slide1R.avi	hand slide ring	drop
247	greenring_20020619_tilt1L.avi	hand tilt ring	
248	greenring_20020619_tilt1R.avi	hand tilt ring	
249	greenring_20020619_tilt2L.avi	hand tilt ring	
250	greenring_20020619_tilt2R.avi	hand tilt ring	drop
251	greenring_20020619_tipover1L.avi	hand tipover ring	
252	greenring_20020619_tipover1R.avi	hand tipover ring	drop
253	greenring_20020619_tipover2L.avi	hand tipover ring	
254	greenring_20020619_tipover2R.avi	hand tipover ring	drop
255	greenring_20020619_touch1L.avi	hand touch ring	
256	greenring_20020619_touch1R.avi	hand touch ring	
257	greenring_20020619_touch2L.avi	hand touch ring	
258	greenring_20020619_touch2R.avi	hand touch ring	
259	greenring_20020619_up1L.avi	hand pickup ring	
260	greenring_20020619_up1R.avi	hand pickup ring	drop
261	greenring_20020619_up2L.avi	hand pickup ring	
262	greenring_20020619_up2R.avi	hand pickup ring	drop
263	greenring_20020619_up3L.avi	hand pickup ring	
264	greenring_20020809_down1L.avi	hand putdown ring	drop
265	greenring_20020809_down1R.avi	hand putdown ring	drop
266	greenring_20020809_down2L.avi	hand putdown ring	drop
267	greenring_20020809_down2R.avi	hand putdown ring	drop
268	greenring_20020809_drop1L.avi	hand drop ring	drop
269	greenring_20020809_drop1R.avi	hand drop ring	drop
270	greenring_20020809_drop2L.avi	hand drop ring	drop
271	greenring_20020809_drop2R.avi	hand drop ring	drop
272	greenring_20020809_drop3L.avi	hand drop ring	drop
273	greenring_20020809_roll1L.avi	hand roll ring	drop

274	greenring_20020809_roll1R.avi	hand roll ring	
275	greenring_20020809_roll2L.avi	hand roll ring	
276	greenring_20020809_roll2R.avi	hand roll ring	drop
277	greenring_20020809_touch1L.avi	hand touch ring	drop
278	greenring_20020809_touch1R.avi	hand touch ring	drop
279	greenring_20020809_touch2L.avi	hand touch ring	drop
280	greenring_20020809_touch2R.avi	hand touch ring	drop
281	greenring_20020809_up1L.avi	hand pickup ring	drop
282	greenring_20020809_up1R.avi	hand pickup ring	drop
283	greenring_20020809_up2L.avi	hand pickup ring	drop
284	greenring_20020809_up2R.avi	hand pickup ring	drop
285	orangecup_20020809_down1L.avi	hand putdown cup	
286	orangecup_20020809_down1R.avi	hand putdown cup	shrunk
287	orangecup_20020809_down2R.avi	hand putdown cup	shrunk
288	orangecup_20020809_pull1L.avi	hand pull cup	drop
289	orangecup_20020809_pull1R.avi	hand pull cup	drop
290	orangecup_20020809_pull2L.avi	hand pull cup	drop
291	orangecup_20020809_pull2R.avi	hand pull cup	drop
292	orangecup_20020809_push1L.avi	hand push cup	drop
293	orangecup_20020809_push1R.avi	hand push cup	drop
294	orangecup_20020809_push2L.avi	hand push cup	drop
295	orangecup_20020809_push2R.avi	hand push cup	drop
296	orangecup_20020809_tip1L.avi	hand tipover cup	split
297	orangecup_20020809_tip1R.avi	hand tipover cup	
298	orangecup_20020809_tip2L.avi	hand tipover cup	split
299	orangecup_20020809_tip2R.avi	hand tipover cup	split
300	orangecup_20020809_touch1L.avi	hand touch cup	
301	orangecup_20020809_touch1R.avi	hand touch cup	
302	orangecup_20020809_touch2L.avi	hand touch cup	
303	orangecup_20020809_touch2R.avi	hand touch cup	
304	orangecup_20020809_up1L.avi	hand pickup cup	
305	orangecup_20020809_up1R.avi	hand pickup cup	
306	orangecup_20020809_up2L.avi	hand pickup cup	
307	orangecup_20020809_up2R.avi	hand pickup cup	
308	rings_20020619_stack1R.avi	hand stack rings	drop
309	rings_20020619_unstack1L.avi	hand unstack rings	drop
310	rings_20020619_unstack1R.avi	hand unstack rings	drop
311	rings_20020619_unstack2R.avi	hand unstack rings	drop
312	vase_20020809_down1L.avi	hand putdown vase	split
313	vase_20020809_down1R.avi	hand putdown vase	
314	vase_20020809_down2L.avi	hand putdown vase	
315	vase_20020809_down2R.avi	hand putdown vase	
316	vase_20020809_push1L.avi	hand push vase	
317	vase_20020809_push1R.avi	hand push vase	split
318	vase_20020809_push2L.avi	hand push vase	
319	vase_20020809_push2R.avi	hand push vase	

320	vase_20020809_touch1L.avi	hand touch vase	
321	vase_20020809_touch1R.avi	hand touch vase	
322	vase_20020809_touch2L.avi	hand touch vase	
323	vase_20020809_touch2R.avi	hand touch vase	
324	vase_20020809_up1L.avi	hand pickup vase	
325	vase_20020809_up1R.avi	hand pickup vase	
326	vase_20020809_up2L.avi	hand pickup vase	
327	vase_20020809_up2R.avi	hand pickup vase	

APPENDIX B – LISTING OF RESOURCES FOR THE EBLA PROJECT

1. <http://www.greatmindsworking.com> - A site dedicated to research on computer models of human language acquisition. It provides a repository of links to relevant news, researchers, conferences, papers, and books from fields including A/I, computational linguistics, developmental psychology, machine learning, and cognitive science. The site also serves as the project site for EBLA.
2. <http://java.sun.com> - Sun Microsystems Java site. Contains news on Java along with links to download the latest Java Software Development Kit (SDK).
3. <http://java.sun.com/j2se/javadoc/index.html> - Sun Microsystems JavaDoc tool homepage.
4. <http://java.sun.com/products/java-media/jmf/> - Sun Microsystems's Java Media Framework API homepage.
5. <http://www.alphaworks.ibm.com/tech/speech> - Web site for the IBM alphaWorks implementation of the Java Speech API based on IBM's ViaVoice technology. It supports voice command recognition, dictation, and text-to-speech synthesis.
6. <http://freetts.sourceforge.net/docs/index.php> - Web site for FreeTTS, an open-source speech synthesizer written entirely in the Java programming language.
7. <http://www.postgresql.org> - Web site for the open-source PostgreSQL relational database.
8. <http://www.pgaccess.org> - Web site for the Linux-based, open-source PGAccess graphical interface for PostgreSQL.
9. <http://pgadmin.postgresql.org> - Web site for the Windows-based, open-source PGAdmin graphical interface for PostgreSQL.
10. <http://jdbc.postgresql.org> - Web site for the PostgreSQL JDBC driver, which allows Java to communicate with a PostgreSQL database.
11. <http://www.virtualdub.org/> - Web site for the open-source VirtualDub video capture and processing software system.
12. <http://www.caip.rutgers.edu/riul/research/code/EDISON/> - Web site for the open-source Edge Detection and Image SegmentatiON (EDISON) software system.
13. <http://www.cvshome.org> - Web site for the open-source Concurrent Versions System (CVS) version control system.
14. <http://www.irfanview.com/> - Web site for the free IrfanView graphic viewer.

APPENDIX C – SAMPLE JAVADOC FOR EBLA

[Overview](#) [Package](#) [Class](#) [Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

com.greatmindsworking.EBLA

Class EBLA

java.lang.Object

|

+--com.greatmindsworking.EBLA.EBLA

```
public class EBLA
    extends java.lang.Object
```

EBLA.java

EBLA - Experience-Based Language Acquisition

This is the main executable class for the EBLA software system.

EBLA is an open computational framework for visual perception and grounded language acquisition. EBLA can watch a series of short videos and acquire a simple language of nouns and verbs corresponding to the objects and object-object relations in those videos. Upon acquiring this protolanguage, EBLA can perform basic scene analysis to generate descriptions of novel videos.

The general architecture of EBLA is comprised of three stages: vision processing, entity extraction, and lexical resolution. In the vision processing stage, EBLA processes the individual frames in short videos, using a variation of the mean shift analysis image segmentation algorithm to identify and store information about significant objects. In the entity extraction stage, EBLA abstracts information about the significant objects in each video and the relationships among those objects into internal representations called entities. Finally, in the lexical acquisition stage, EBLA extracts the individual lexemes (words) from simple descriptions of each video and attempts to generate entity-lexeme mappings using an inference technique called cross-situational learning. EBLA is not primed with a base lexicon, so it faces the task of bootstrapping its lexicon from scratch.

While there have been several systems capable of learning object or event labels for videos, EBLA is the first known system to acquire both nouns and verbs using a grounded computer vision system.

EBLA was developed as part of Brian E. Pangburn's dissertation research in the Department of Computer Science at Louisiana State University.

The full dissertation along with other information on EBLA is available from <http://www.greatmindsworking.com>

TO-DO:

1. load database parameters (DB name, IP, username, pwd) from a text file
2. finish code for dynamic loading of attribute calculations using forName()
3. revisit color attributes (R, G, B) and color reduction
4. weight words based on # of prior occurrences when generating descriptions

Version:

\$Revision: 1.24 \$ \$Date: 2002/10/02 20:50:52 \$

Author:

\$Author: bpanburn \$

Constructor Summary

<u>EBLA</u> ()
Class constructor that initializes database connection and sets the runtime parameters based on the defaults in the Params class
<u>EBLA</u> (long _parameterID)
Class constructor that initializes database connection and looks up runtime parameters based on the user specified parameter ID

Method Summary

void	<u>dispose</u> (boolean _saveChanges)	Closes database connection and sets objects that are no longer needed to null
static void	<u>main</u> (java.lang.String[] args)	Main procedure - allows EBLA to be run from the command line.
void	<u>processExperiences</u> ()	Performs video processing, entity extraction, and lexical resolution for a set of experiences.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EBLA

```
public EBLA(long _parameterID)
```

Class constructor that initializes database connection and looks up runtime parameters based on the user specified parameter ID

Parameters:

_parameterID - long containing database record id for runtime parameters (-1 if unavailable)

EBLA

public **EBLA**()

Class constructor that initializes database connection and sets the runtime parameters based on the defaults in the Params class

Method Detail

processExperiences

public void **processExperiences**()

Performs video processing, entity extraction, and lexical resolution for a set of experiences.

The subset of experiences to be processed is specified by the Params class. The experiences will be processed a set number of times for a range of minimum standard deviation values. These minimum standard deviation values determine how closely the attribute values for two objects or object-object relations must match for them to be considered instances of the same entity. The starting and stopping minimum standard deviation values along with the step size and number of iterations are all specified in the Params class.

The video processing, entity extraction, and lexical resolution phases of EBLA can be executed independently based on boolean flags in the Params class. This allows the computationally expensive video processing phase to be run separately from the other phases.

dispose

public void **dispose**(boolean _saveChanges)

Closes database connection and sets objects that are no longer needed to null

Parameters:

`_saveChanges` - boolean indicating whether or not to save database changes

main

```
public static void main(java.lang.String[] args)
```

Main procedure - allows EBLA to be run from the command line.

The user can pass a parameter ID from the command line to specify a set of EBLA parameters in the `parameter_data` table in the `ebla_data` database. Otherwise EBLA will initialize without a parameter ID and use the hard-coded default parameters in the `Params` class.

[Overview](#) [Package](#) **[Class](#)** [Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

APPENDIX D – SQL USED TO CONSTRUCT THE EBLA DATABASE

```
/* $Id: ebla_data.sql,v 1.8 2002/10/09 01:49:53 bpangburn Exp $
*
* Tab Spacing = 4
*
* Copyright (c) 2002, Brian E. Pangburn
* All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are met:
*
* Redistributions of source code must retain the above copyright notice, this
* list of conditions and the following disclaimer.  Redistributions in binary
* form must reproduce the above copyright notice, this list of conditions and
* the following disclaimer in the documentation and/or other materials
* provided with the distribution.  The names of its contributors may not be
* used to endorse or promote products derived from this software without
* specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*/

/* This SQL script is used to generate the database for the ebla_data
   database in Postgres */

/* 1st-level Parameter Data table */

/* This table is used to initialize the run-time parameters for EBLA */

CREATE SEQUENCE parameter_data_seq;
CREATE TABLE parameter_data (
  /* UNIQUE ID FOR EACH PARAMETER RECORD */
  parameter_id          INTEGER          PRIMARY KEY
                        DEFAULT nextval('parameter_data_seq'),
  /* DESCRIPTION OF PARAMETER SET */
  description           VARCHAR(100),
  /* MATCHED TO INCLUDE_CODE IN EXPERIENCE_DATA TO DETERMINE WHICH
     EXPERIENCES TO PROCESS */
  include_code          INT2            DEFAULT 1
                        NOT NULL,
  /* 0=NO; 1=YES (PROCESS VIDEOS UP TO ENTITY EXTRACTION STAGE) */
  process_videos_code   INT2            DEFAULT 0
                        NOT NULL,
  /* 0=NO; 1=YES (PROCESS ENTITIES) */
  process_entities_code INT2            DEFAULT 0
                        NOT NULL,
  /* 0=NO; 1=YES (PROCESS LEXEMES) */
  process_lexemes_code  INT2            DEFAULT 0
                        NOT NULL,
);
```

```

/* 0=NO; 1=YES (REDIRECT SCREEN OUTPUT TO LOG FILE) */
log_to_file_code      INT2          DEFAULT 0
                        NOT NULL,
/* 0=NO; 1=YES (RANDOMIZE EXPERIENCES WHEN QUERYING FROM DATABASE) */
randomize_exp_code    INT2          DEFAULT 0
                        NOT NULL,
/* 0=NO; 1=YES (ATTEMPT TO GENERATE DESCRIPTIONS FOR SOME EXPERIENCES) */
generate_desc_code    INT2          DEFAULT 0
                        NOT NULL,
/* NUMBER OF EXPERIENCES TO PROCESS BEFORE GENERATING DESCRIPTIONS */
desc_threshold        INTEGER       DEFAULT 7
                        NOT NULL,
/* STARTING MINIMUM STANDARD DEVIATION */
min_sd_start          INTEGER       DEFAULT 5
                        NOT NULL,
/* STOPPING MINIMUM STANDARD DEVIATION */
min_sd_stop           INTEGER       DEFAULT 5
                        NOT NULL,
/* MINIMUM STANDARD DEVIATION STEP SIZE*/
min_sd_step           INTEGER       DEFAULT 5
                        NOT NULL,
/* # OF TIMES TO PROCESS EXPERIENCES FOR EACH MIN STANDARD DEVIATION */
loop_count            INTEGER       DEFAULT 1
                        NOT NULL,
/* 0=NO; 1=YES (LIMIT STANDARD DEVIATION FOR ENTITY MATCHING TO SPECIFIED
VALUE - IF NO THEN UTILIZE CALCULATED SD FOR CURRENT ENTITY
ATTRIBUTES) */
fixed_sd_code         INT2          DEFAULT 0
                        NOT NULL,
/* PATH TO STORE TEMPORARY FILES DURING PROCESSING */
tmp_path              VARCHAR(50),
/* 0=NO; 1=YES (DISPLAY MOVIE DURING FRAME EXTRACTION) */
display_movie_code    INT2          DEFAULT 0
                        NOT NULL,
/* 0=NO; 1=YES (DISPLAY DETAILED DATA DURING FRAME PROCESSING) */
display_text_code     INT2          DEFAULT 0
                        NOT NULL,
/* 0=NO; 1=YES (SAVE TEMP IMAGE FILES AFTER PROCESSING) */
save_images_code      INT2          DEFAULT 0
                        NOT NULL,
/* COLOR RADIUS FOR MEAN-SHIFT ANALYSIS COLOR IMAGE SEGMENTATION */
seg_color_radius      FLOAT         DEFAULT 6.5
                        NOT NULL,
/* SPATIAL RADIUS FOR MEAN-SHIFT ANALYSIS COLOR IMAGE SEGMENTATION */
seg_spatial_radius   INTEGER       DEFAULT 7
                        NOT NULL,
/* MINIMUM PIXEL REGION FOR MEAN-SHIFT ANALYSIS COLOR IMAGE SEGMENTATION */
seg_min_region        INTEGER       DEFAULT 20
                        NOT NULL,
/* 0=NO SPEEDUP; 1=MEDIUM SPEEDUP; 2=HIGH SPEEDUP (SPEEDUP FOR
MEAN-SHIFT ANALYSIS COLOR IMAGE SEGMENTATION */
seg_speed_up_code     INT2          DEFAULT 0
                        NOT NULL,
/* FILE PREFIX FOR TEMP FRAMES EXTRACTED FROM EACH MOVIE/EXPERIENCE */
frame_prefix          VARCHAR(50),
/* FILE PREFIX FOR TEMP SEGMENTED IMAGES CREATED FOR EACH FRAME */
seg_prefix            VARCHAR(50),
/* FILE PREFIX FOR TEMP POLYGON IMAGES CREATED FOR EACH FRAME */
poly_prefix           VARCHAR(50),
/* PERCENTAGE OF TOTAL PIXELS THAT AN OBJECT MUST CONTAIN TO BE CONSIDERED
PART OF THE BACKGROUND RATHER THAN A SIGNIFICANT OBJECT (0 - 100) */
background_pixels     FLOAT         DEFAULT 20.0
                        NOT NULL,

```

```

/* MINIMUM NUMBER OF PIXELS THAT CONSTITUTE A "SIGNIFICANT" OBJECT */
min_pixel_count          INTEGER          DEFAULT 500
                                NOT NULL,
/* MINIMUM NUMBER OF CONSECUTIVE FRAMES THAT AN OBJECT MUST APPEAR IN TO
   BE CONSIDERED A SIGNIFICANT OBJECT (HELPS TO ELIMINATE NOISE /
   SHADOWS). */
min_frame_count          INTEGER          DEFAULT 7
                                NOT NULL,
/* 0=NO; 1=YES (REDUCE COLOR DEPTH OF SEGMENTED REGIONS) */
reduce_color_code        INT2            DEFAULT 0
                                NOT NULL,
/* 0=NO; 1=YES (LEXEMES ARE CASE-SENSITIVE) */
case_sensitive_code      INT2            DEFAULT 0
                                NOT NULL,
/* NOTES ABOUT THE PARAMETERS */
notes                    VARCHAR(255)

);

/* 1st-level Experience Data table */

/* This table contains information about the multimedia file representing an
   EBLA perceptual experience along with a description of the experience */

CREATE SEQUENCE experience_data_seq;
CREATE TABLE experience_data (
/* UNIQUE ID FOR EACH EXPERIENCE DATA RECORD */
experience_id            INTEGER          PRIMARY KEY
                                DEFAULT nextval('experience_data_seq'),
/* INTERNAL NAME OF EXPERIENCE */
name                    VARCHAR(50) NOT NULL,
/* ORDER THAT EXPERIENCE IS PROCESSED (USED FOR DETERMINING HOW LONG IT
   TAKES TO RESOLVE EACH LEXEME */
experience_index         INTEGER          DEFAULT 0
                                NOT NULL,
/* COMPLETE PATH AND FILENAME OF SOURCE MOVIE (AVI OR MOV) THAT CONTAINS
   EXPERIENCE */
source_movie            VARCHAR(100)     NOT NULL,
/* SHORT DESCRIPTION OF EXPERIENCE (NO SPACES) - USED TO CREATE PATH FOR
   INTERMEDIATE RESULTS */
label                  VARCHAR(50) NOT NULL,
/* 0=NO; 1=YES (GENERATE LANGUAGE DESCRIBING EVENT BASED ON INTERNAL
   LEXICON) */
generate_code           INT2            DEFAULT 0
                                NOT NULL,
/* LANGUAGE TEXT ASSOCIATED WITH EXPERIENCE (SUPPLIED) */
language_text          VARCHAR(100),
/* PROTOLANGUAGE GENERATED BY EBLA TO DESCRIBE AN EXPERIENCE BASED ON
   PRIOR EXPERIENCES */
description            VARCHAR(100),
/* 0=NO; 1=YES (INCLUDE EXPERIENCE IN CURRENT PROCESSING) */
include_code           INT2            DEFAULT 1
                                NOT NULL,
/* NOTES ABOUT THE EXPERIENCE */
notes                  VARCHAR(255)

);
CREATE INDEX exp_name_idx ON experience_data (name);

/* 1st-level Attribute List Data table */

```

```
/* This table contains a list of the attributes that can be detected by EBLA */
```

```
CREATE SEQUENCE attribute_list_data_seq;
CREATE TABLE attribute_list_data (
  /* UNIQUE ID FOR EACH ATTRIBUTE LIST DATA RECORD */
  attribute_list_id      INTEGER      PRIMARY KEY
                          DEFAULT
                          nextval('attribute_list_data_seq'),

  /* NAME OF ATTRIBUTE */
  name                   VARCHAR(50) NOT NULL,
  /* 0=NO; 1=YES (INCLUDE ATTRIBUTE WHEN ANALYZING EXPERIENCES) */
  include_code           INT2         DEFAULT 1
                          NOT NULL,
  /* 0=OBJECT; 1=RELATION (INDICATES WHETHER ATTRIBUTE APPLIES TO AN OBJECT
  OR THE RELATION BETWEEN TWO OBJECTS) */
  type_code              INT2         DEFAULT 0
                          NOT NULL,
  /* NAME OF JAVA CLASS THAT SHOULD BE INVOKED TO ANALYZE ATTRIBUTE
  (CURRENTLY NOT IMPLEMENTED) */
  class_name             VARCHAR(50),
  /* NOTES ABOUT THE EXPERIENCE */
  notes                  VARCHAR(255)
);
CREATE INDEX att_lis_name_idx ON attribute_list_data (name);
```

```
/* 1st-level Entity Data table */
```

```
/* This table contains all of the entities that have been detected by EBLA */
```

```
CREATE SEQUENCE entity_data_seq;
CREATE TABLE entity_data (
  /* UNIQUE ID FOR EACH ENTITY DATA RECORD */
  entity_id              INTEGER      PRIMARY KEY
                          DEFAULT nextval('entity_data_seq'),
  /* NUMBER OF TIMES THAT ENTITY HAS BEEN RECOGNIZED WHEN PROCESSING EBLA
  EXPERIENCES */
  occurrence_count       INTEGER      DEFAULT 1
                          NOT NULL
);
```

```
/* 1st-level Lexeme Data table */
```

```
/* This table contains all of the lexical items that have been detected by
EBLA. A lexical item can occur in the table multiple times if multiple
senses of the word are encountered */
```

```
CREATE SEQUENCE lexeme_data_seq;
CREATE TABLE lexeme_data (
  /* UNIQUE ID FOR EACH LEXICAL ITEM RECORD */
  lexeme_id             INTEGER      PRIMARY KEY
                          DEFAULT nextval('lexeme_data_seq'),
  /* LEXICAL ITEM / WORD */
  lexeme                VARCHAR(50) NOT NULL,
  /* NUMBER OF TIMES THAT LEXICAL ITEM HAS BEEN RECOGNIZED WHEN PROCESSING
  EBLA EXPERIENCES */
  occurrence_count      INTEGER      DEFAULT 1
                          NOT NULL
);
```

```

);
CREATE INDEX lex_lexeme_idx ON lexeme_data (lexeme);

/* 2nd-level Frame Analysis Data table */

/* This table contains the preliminary information about the "significant"
objects encountered in an EBLA experience */

CREATE SEQUENCE frame_analysis_data_seq;
CREATE TABLE frame_analysis_data (
/* UNIQUE ID FOR EACH FRAME ANALYSIS RECORD */
frame_analysis_id      INTEGER      PRIMARY KEY
                        DEFAULT
                        nextval('frame_analysis_data_seq'),

/* ID OF PARENT EXPERIENCE RECORD */
experience_id          INTEGER      REFERENCES experience_data
                        ON UPDATE CASCADE
                        ON DELETE CASCADE,

/* NUMBER OF CURRENT FRAME */
frame_number           INTEGER      DEFAULT 0
                        NOT NULL,

/* OBJECT INDEX (AS OBJECTS ARE DETECTED, THEY ARE INDEXED FROM THE TOP
DOWN AND CORRELATED FROM FRAME TO FRAME) */
object_number          INTEGER      DEFAULT 0
                        NOT NULL,

/* NUMBER OF POINTS IN POLYGON */
polygon_point_count    INTEGER      DEFAULT 0
                        NOT NULL,

/* COMMA-SEPARATED LIST OF POLYGON POINTS */
polygon_point_list     TEXT         NOT NULL,

/* RGB COLOR OF OBJECT (27 POSSIBLE VALUES - EACH RGB COMPONENT IS ROUNDED
TO 0, 128, OR 255) */
rgb_color              INTEGER      DEFAULT 0
                        NOT NULL,

/* COMMA-SEPARATED LIST OF BOUNDING RECTANGLE POINTS */
bound_rect_points      VARCHAR(50) NOT NULL,

/* X COORDINATE OF CENTER OF GRAVITY */
centroid_x             INTEGER      DEFAULT 0
                        NOT NULL,

/* Y COORDINATE OF CENTER OF GRAVITY */
centroid_y             INTEGER      DEFAULT 0
                        NOT NULL,

/* AREA OF OBJECT */
area                   FLOAT        DEFAULT 0
                        NOT NULL
);
CREATE INDEX fra_ana_experience_id_idx ON frame_analysis_data (experience_id);
CREATE INDEX fra_ana_frame_number_idx ON frame_analysis_data (frame_number);
CREATE INDEX fra_ana_object_number_idx ON frame_analysis_data (object_number);

/* 2nd-level Attribute-Value Data table */

/* This table contains the attribute values for each entity encountered
in an EBLA experience */

CREATE SEQUENCE attribute_value_data_seq;
CREATE TABLE attribute_value_data (
/* UNIQUE ID FOR EACH ATTRIBUTE-VALUE RECORD */
attribute_value_id     INTEGER      PRIMARY KEY

```

```

                                DEFAULT
                                nextval('attribute_value_data_seq'),
/* ID OF PARENT ATTRIBUTE LIST DATA RECORD */
attribute_list_id      INTEGER      REFERENCES attribute_list_data
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
/* ID OF PARENT ENTITY DATA RECORD */
entity_id              INTEGER      REFERENCES entity_data
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
/* AVERAGE VALUE OF ATTRIBUTE */
avg_value              FLOAT        DEFAULT 0
                                NOT NULL,
/* STANDARD DEVIATION OF ATTRIBUTE */
std_deviation          FLOAT        DEFAULT 0
                                NOT NULL
);
CREATE INDEX att_val_attribute_list_id_idx ON attribute_value_data
(attribute_list_id);
CREATE INDEX att_val_entity_id_idx ON attribute_value_data (entity_id);

/* 2nd-level Experience-Entity Data table */

/* This table contains a record with the ID of each entity record in an EBLA
experience */

CREATE SEQUENCE experience_entity_data_seq;
CREATE TABLE experience_entity_data (
/* UNIQUE ID FOR EACH EXPERIENCE-ENTITY RECORD */
experience_entity_id  INTEGER      PRIMARY KEY
                                DEFAULT
                                nextval('experience_entity_data_seq'),
/* ID OF PARENT EXPERIENCE DATA RECORD */
experience_id          INTEGER      REFERENCES experience_data
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
/* ID OF PARENT ENTITY DATA RECORD */
entity_id              INTEGER      REFERENCES entity_data
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
/* 0=NO; 1=YES (CODE INDICATING IF ENTITY HAS BEEN RESOLVED TO A LEXICAL
ITEM) */
resolution_code        INT2        DEFAULT 0
                                NOT NULL
);
CREATE INDEX exp_ent_experience_id_idx ON experience_entity_data
(experience_id);
CREATE INDEX exp_ent_entity_id_idx ON experience_entity_data (entity_id);

/* 2nd-level Experience-Lexeme Data table */

/* This table contains a record with the ID of each lexeme record in an EBLA
experience */

CREATE SEQUENCE experience_lexeme_data_seq;
CREATE TABLE experience_lexeme_data (
/* UNIQUE ID FOR EACH EXPERIENCE-LEXEME RECORD */
experience_lexeme_id  INTEGER      PRIMARY KEY
                                DEFAULT

```

```

                                nextval('experience_lexeme_data_seq'),
/* ID OF PARENT EXPERIENCE DATA RECORD */
experience_id          INTEGER      REFERENCES experience_data
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
/* ID OF PARENT LEXEME DATA RECORD */
lexeme_id             INTEGER      REFERENCES lexeme_data
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
/* 0=NO; 1=YES (CODE INDICATING IF ENTITY HAS BEEN RESOLVED TO A LEXICAL
ITEM) */
resolution_code       INT2         DEFAULT 0
                                NOT NULL,
/* NUMBER OF EXPERIENCES PROCESSED BEFORE RESOLUTION OCCURS (ZERO IF NOT
RESOLVED) */
resolution_index      INTEGER      DEFAULT 0
                                NOT NULL
);
CREATE INDEX exp_lex_experience_id_idx ON experience_lexeme_data
(experience_id);
CREATE INDEX exp_lex_lexeme_id_idx ON experience_lexeme_data (lexeme_id);

/* 2nd-level Entity-Lexeme Data table */

/* This table contains a record with the entity-lexeme mappings */

CREATE SEQUENCE entity_lexeme_data_seq;
CREATE TABLE entity_lexeme_data (
/* UNIQUE ID FOR EACH ENTITY-LEXEME RECORD */
entity_lexeme_id      INTEGER      PRIMARY KEY
                                DEFAULT
                                nextval('entity_lexeme_data_seq'),
/* ID OF PARENT ENTITY DATA RECORD */
entity_id             INTEGER      REFERENCES entity_data
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
/* ID OF PARENT LEXEME DATA RECORD */
lexeme_id             INTEGER      REFERENCES lexeme_data
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
/* NUMBER OF TIMES THAT LEXICAL ITEM HAS BEEN RECOGNIZED WHEN PROCESSING
EBLA EXPERIENCES */
occurance_count       INTEGER      DEFAULT 1
                                NOT NULL
);
CREATE INDEX ent_lex_entity_id_idx ON entity_lexeme_data (entity_id);
CREATE INDEX ent_lex_lexeme_id_idx ON entity_lexeme_data (lexeme_id);

```

VITA

Brian Edward Pangburn was born in Melrose, Massachusetts, on June 12, 1972. He received his bachelor of science degree in mechanical engineering from Tulane University in 1994. He joined the doctoral program at Louisiana State University in 1994, and received his master of science degree in system science in 1999. He will receive the degree of Doctor of Philosophy in computer science at the December, 2002 commencement.

Since 1991, Mr. Pangburn has developed software to administer retirement plans. From 1994 to 1996, he developed software as a consultant for American Financial Systems in Weston, Massachusetts. In 1996, he co-founded The Pangburn Company, a third party administration company dealing exclusively with nonqualified deferred compensation retirement plans.

Apart from language acquisition modeling, his research interests include artificial intelligence, computer vision, and database theory.

Mr. Pangburn has a wife, Jaimee, and twin sons, Jack and Jeremiah. He and his family reside in Ventress, Louisiana.