

©Copyright 2008
Margaret A. Mitchell

Towards the Generation of Natural Reference

Margaret A. Mitchell

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Arts

University of Washington

2008

Program Authorized to Offer Degree: Department of Linguistics

University of Washington
Graduate School

This is to certify that I have examined this copy of a master's thesis by

Margaret A. Mitchell

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Scott Farrar

Emily M. Bender

Date: _____

In presenting this thesis in partial fulfillment of the requirements for a master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purpose or by any means shall not be allowed without my written permission.

Signature_____

Date_____

University of Washington

Abstract

Towards the Generation of Natural Reference

Margaret A. Mitchell

Chair of the Supervisory Committee:
Professor Scott Farrar
Computational Linguistics

In this thesis, I examine the task of generating referring expressions that sound *natural*, mirroring reference produced by people. I focus on psychological and semantic issues to formulate a computational model of natural human reference. The algorithm I develop is shown to be capable of generating most referring expressions elicited from human participants in an evaluation task.

To create a computational model of natural reference, I separate two forms of reference: reference to distinguish an entity, and reference to describe an entity. Reference to distinguish an entity is generated by the algorithm I develop. The algorithm contains an approach to prenominal modifier ordering that may be used for either form.

This algorithm is composed of three major functions, paralleling the three components common to natural language generation systems. The content determination function derives two types of modifiers, using an incremental process to derive modifiers from comparison of the intended referent to other items in the context. The microplanning function selects specific modifiers to be used, creating a structure that maps directly to a single referring expression. A type system based on modifier distribution is developed for the surface realization stage of the algorithm, and is used to order modifiers prenominally.

The type system is extensible to a variety of applications, and I evaluate the system on a corpus. It is shown to be robust and accurate. The algorithm is evaluated against human data and shown to successfully reflect natural language use.

TABLE OF CONTENTS

	Page
List of Figures	iii
List of Tables	iv
Glossary	v
Chapter 1: Introduction	1
1.1 Background	2
1.2 Naturalness	4
1.3 The Approach	8
1.4 Thesis Organization	12
Chapter 2: Literature Review	13
2.1 The Beginning	13
2.2 The Field of Natural Language Generation Begins	15
2.3 Referring Expression Generation	16
2.4 What Psychological Evidence Has to Say	21
2.5 Improvements Since the Introduction of the Incremental Algorithm	22
2.6 Prenominal Ordering of Adjectives	25
2.7 Summary	26
Chapter 3: The Algorithm	28
3.1 Overview	28
3.2 Main Ideas	28
3.3 Towards Generation	30
3.4 Overview of the Algorithm	32
3.5 Inputs and Outputs	34
3.6 System Knowledge	35
3.7 The Referent Vector and the Contrast Vectors	36

3.8	The Algorithm	41
3.9	Algorithm Summary	46
3.10	Example	47
3.11	Example Implementation	49
3.12	Discussion	51
3.13	Summary	52
Chapter 4:	Ordering Modifiers for Referents	53
4.1	Introduction	53
4.2	The Model and the Problem	54
4.3	Towards a Solution	55
4.4	Materials	58
4.5	Method	60
4.6	Results	64
4.7	Discussion	65
Chapter 5:	Evaluation	68
5.1	Introduction	68
5.2	Purpose	68
5.3	Materials	69
5.4	Method	69
5.5	Results	75
5.6	Discussion	76
Chapter 6:	Conclusions	79
	Bibliography	80
Appendix A:	Pictures from Evaluation	88
Appendix B:	Filtered Modifiers for the Modifier Classification System	91
Appendix C:	Contrast Set Mappings	92
Appendix D:	Mechanical Turk	93

LIST OF FIGURES

Figure Number	Page
1.1 Distinguishing Reference and Describing Reference	9
1.2 Example Distinguishing Reference	11
2.1 The Gricean Maxims	14
2.2 The Incremental Algorithm	20
3.1 Flow Chart of Algorithm for Distinguishing Reference	29
3.2 Example Input Domain	37
3.3 Example Intended Referent Vector	37
3.4 Example Contrast Set Vectors	37
3.5 Example Scene: a Pig, a Duck, a Mouse, and a Horse	39
3.6 Text Planning: Algorithm for Distinguishing Reference	42
3.7 Microplanning: Algorithm for Distinguishing Reference	43
3.8 Text Planning Example Referring Expression Structure	48
3.9 Microplanning Example Referring Expression Structures	49
3.10 Example Generated Referring Expressions	49
3.11 Example ILEX Input	49
3.12 Speech Act Input for WAG (O'Donnell, 2006: 7)	50
3.13 Identifiable Entity for Surface Realization	51
4.1 Grammatical Adjective Alternations (Vendler, 1968: 122)	54
4.2 Example 5-Gram with Prenominal Positions	60
D.1 Mechanical Turk: Screenshot	93

LIST OF TABLES

Table Number	Page
1.1 Natural Language Generation: Outline (Reiter and Dale, 2000)	3
3.1 Example Input-Output Pairs	34
4.1 Example Modifier Classification Intermediate File: Step 3	57
4.2 Example Modifier Classification Intermediate File: Step 4	57
4.3 Modifier Types	61
4.4 Frequency of Prenominal Modifier Amounts	61
4.5 Modifier Type Distribution	62
4.6 Proposed Modifier Ordering	63
4.7 Precision and Recall for Prenominal Modifier Ordering	65
5.1 Incremental Algorithm Constraints	74
5.2 Proposed Algorithm Constraints	74
5.3 Natural Variation in Referring Expression Choice	75
5.4 Algorithm Results	76
B.1 Filtered Modifiers in Modifier Type System	91
C.1 Example Contrast Set Properties: Size Relations and Corresponding Adjectives	92

GLOSSARY

CONTEXT SET: set of entities

CONTRAST ITEM: entity not being referred to

CONTRAST SET: set of entities excluding the one being referred to

DESCRIBING REFERENCE: phrase with modifiers used to describe an entity

DISTINGUISHING DESCRIPTION: set of attribute-value pairs that uniquely identify a referent

DISTINGUISHING REFERENCE: phrase with modifiers used to distinguish an entity

INPUT DOMAIN: context from which a referring expression is generated

INTENDED REFERENT: entity being referred to

KB: Knowledge Base

LOOSE CONTRADICTION: relationship between two words where one entails or does not entail the other

NLG: Natural Language Generation

NLP: Natural Language Processing

NP: Noun Phrase

POS: Part of Speech

PRENOMINAL: occurring immediately before a noun

REFERRING EXPRESSION: phrase used to refer to something

STRICT CONTRADICTION: antonym relationship between two words

ACKNOWLEDGMENTS

I am grateful to everyone who has brought me to this point. Whether this marks the beginning or the end of my research on reference, I have arrived here by the diverse influences of all those around me. Thanks to my advisors, Scott Farrar and Emily Bender, for keeping their eyes on the end goal and working with me towards coherent conclusion. Thanks to my mom, for the cards and fun doodads and unshakable belief that whatever I do is golden. Thanks to Robert Dale and Jette Viethen for words of wisdom and encouragement, and for making me feel welcome in the NLG crowd. Thanks to my fellow students in the CLMA program, who grew from strangers to some of my dearest friends over this rapid and intense year. Thanks to the people in Salem for keeping me grounded. And thanks to my fiancé and best friend, Scott, whose strength, self-assurance, and confidence in me gave me the determination to strive for academic integrity. Also thanks to Scott for making me all that coffee. That was awesome.

DEDICATION

To fortitude

Chapter 1

INTRODUCTION

Referring is the task of using language to identify or pick out an entity. **Natural language generation**, the task of generating language from a machine representation of knowledge, utilizes **referring expression generation** algorithms to produce expressions that refer. Algorithms for the generation of referring expressions aim to produce descriptions of entities as quickly as possible, as effectively as possible, and as naturally as possible.

In this thesis, I introduce an algorithm for the generation of referring expressions. This algorithm is created to generate expressions that sound *natural*. I build on previous work to develop an algorithm capable of generating the same kinds of expressions that people use. The algorithm is shown to be capable of generating referring expressions identical to the vast majority of human-produced referring expressions in a visual presentation task, and is more successful at creating natural reference than one of the most prominent algorithms in the field, the Incremental Algorithm (Dale and Reiter, 1995).

This chapter provides an introduction to the generation of natural referring expressions. In the next section, I explain how the generation of referring expressions is placed within the broader context of natural language generation.¹ My discussion of generation is followed by an analysis of what it means for an expression to be natural, and how naturalness can be modelled in an algorithm that generates referring expressions. Emphasis is placed specifically on the Gricean maxims and Thomas Pechmann’s idea of incremental speech production, which have been used to guide naturalness in the past. A summary of these ideas and a basic outline of the algorithm follows the discussion of naturalness. The chapter

¹Throughout this thesis, the term “natural language generation” is used to refer to the process of generating natural language, while the term “Natural Language Generation” is used to refer specifically to the field exploring the issue. I also use this convention to separate the field of “Referring Expression Generation” from the task of “referring expression generation”, and to separate the broad components of an NLG system from the tasks therein.

concludes with an overview of the structure of the thesis as a whole.

1.1 Background

Previous research on the generation of referring expressions has largely aimed at producing a **distinguishing description** of a target object, a set of attribute-value pairs which are together true of the intended referent but of no other entity in the context set (Dale, 1989; see Chapter 2 for more). Yet even for the generation of specific, distinguishing reference to entities, formulating an algorithm is difficult. Work in this area has focused on a range of different goals, with vastly different results. These goals can be broadly categorized as prioritizing among the following three objectives:

- Creating a computationally efficient algorithm
- Creating an algorithm that uniquely identifies a referent (the goal-driven approach)
- Creating an algorithm that produces expressions reflecting psycholinguistic realities of human reference

Algorithms are not limited to just one of these goals, and the most prominent algorithm in the generation of referring expressions, Dale and Reiter’s Incremental Algorithm (Dale and Reiter, 1995), produces expressions motivated by all three. But there is a trade-off between these three goals, and it is difficult to fully realize any one of them without compromising the other two. The extent of the trade-off depends on the nature of the system. Algorithms for the generation of referring expressions have largely prioritized the first two goals (Areces et al., 2008; Gardent, 2002). This thesis introduces an algorithm that prioritizes the third.

To simplify the problem, I only approach the generation of expressions headed by a noun and preceded by a series of modifiers, such as those used for first reference to an entity. Examples include *the tall man* and *an icy torrential rain*. I exclude in this grouping any phrases with embedded prepositional phrases, complementizer phrases, or relative clauses. The issue of definiteness is outside the scope of this work as well, and the selection of articles will not be addressed.

The algorithm I present differs from previous approaches in that it does not create distinguishing descriptions of items. A distinguishing description is a goal-driven logical conjunction of properties. A distinguishing description must, by definition, uniquely identify an intended referent, which makes it an unsuitable term for the model of natural language I develop. Below, I define separate terms for reference that are specifically aimed at capturing natural reference.

I follow previous work and define the entity to be picked out from a group of entities as the **intended referent**, the group in which it exists as the **context set**, and the group of entities not including the intended referent as the **contrast set**. These items form the basis of most referring expression generation algorithms.

The algorithm developed in this thesis is composed of three major functions, which may be incorporated into the three stages common to natural language generation systems. These three stages are referred to as Text Planning, Microplanning, and Surface Realization. An outline of this process is given in Table 1.1. This outline and further discussion of its parts can be found in (Reiter and Dale, 2000).

Table 1.1: Natural Language Generation: Outline (Reiter and Dale, 2000)

Module	Content task	Structure task
Text Planner	content determination	discourse planning
Microplanner	lexicalization, referring expression generation	aggregation
Surface Realizer	linguistic realization	structure realization

The **Text Planner** is the module that selects the information that will be expressed by the system. The **Microplanner** converts this information into linguistic form, and is traditionally where referring expression generation algorithms are incorporated. The **Surface Realizer** generates the linguistic forms as readable (or audible) sentences. This is a general categorization, and the specifics of each module depend on the generation system.

With the task of referring expression generation defined and placed within the broader context of natural language generation, I now turn to an analysis of what it means for reference to be natural. This allows me to begin formulating a computational model of natural referring expression generation.

1.2 *Naturalness*

Two major sources are widely used to model naturalness in referring expression generation: the **Gricean maxims** and **incremental speech production**. The Gricean maxims have dominated the field as principles to guide naturalness, and I explain why these maxims cannot be used when the primary goal is to generate natural language. The theory of incremental speech production is incorporated into my own algorithm, and I analyze how my interpretation of this process differs from how it is commonly interpreted. I then introduce the outline of my algorithm.

1.2.1 *The Gricean Maxims, Underspecification, and Overspecification*

In the absence of concrete rules that define language naturalness, workers in Natural Language Generation turn to H. P. Grice (Grice, 1975; see Dale and Reiter, 1995; Sripada and Reiter, 2003). Grice developed maxims in an attempt to fit metaphysical issues into the language of logic. His maxims have been adopted by those working in NLG as a list of guidelines people use when generating language. The Gricean maxims and a detailed discussion of Grice’s work are available in Chapter 2.

Grice believed that the maxims are those which participants “will be expected” to observe in conversation. Grice does not make the argument that the maxims capture what people do when they communicate, and instead comments that these maxims may be used to help guide the “maximally effective exchange of information” (58). The goal of the maxims is not to guide naturalness, but rather to guide a kind of idealized conversation. Utilizing these maxims to guide naturalness therefore presents a problem. *What people do* when they speak – the ultimate measure of what is *natural* – is often quite different than what the maxims capture.

Language produced by people differs from what is captured by the Gricean maxims in two clear ways: Expressions are often **underspecified**, failing to distinguish the intended referent (Clark and Wilkes-Gibbs, 1985; Pechmann, 1989; Viethen and Dale, 2008), and even more commonly are **overspecified**, with descriptors that rule out no members of the contrast set (Sonnenschein, 1985). This use of *overspecified* follows the definition originally used by Pechmann (1989), to denote an expression that contains features (or properties) which are *not distinguishing at all* (98).

Because the Gricean maxims fail to capture what naturally happens in language, they cannot be used to guide the generation of natural language. Instead, a model that accounts for what people do when they speak, including underspecification and overspecification, must be formalized. This is developed by utilizing natural language corpora and the ideas behind incremental speech production.

1.2.2 Incremental Speech Production

The Incremental Algorithm and other algorithms informed by the Gricean maxims seek to emulate natural language by modelling Thomas Pechmann’s observation that people produce utterances incrementally (Pechmann, 1989). The research provided by Pechmann is illuminating, and extremely useful for formulating computational models of human language. However, how it is commonly incorporated is not reflective of Pechmann’s findings. Pechmann’s study is detailed in Chapter 2, and I will only discuss it briefly here before examining how the findings are implemented in my own approach.

In Pechmann’s study, participants wearing eye-trackers were presented with pictures of objects that varied in size and color. They were asked to name one object so that the experimenter “unambiguously knew which object was meant” (94). Pechmann found that as his participants named a target object, they observed the other surrounding objects. The idea Pechmann developed to account for this is that each object that is fixated on helps determine what is next said to identify the target object. This is the idea of **incremental speech production**.

As described by Pechmann, incremental speech production has two major components:

1. A separation between two kinds of properties
2. An incremental analysis of the surrounding objects (the contrast set) to derive descriptors for one kind of property

Pechmann's findings have generally been implemented as an incremental process that operates on attribute-value pairs (not the contrast set) that hold for the entire context set (not just one type of property). My algorithm adopts a more straightforward analysis of Pechmann's findings. I will now discuss each component in greater detail.

Two Kinds of Properties

I follow Pechmann in separating out two major kinds of properties to pick out an item from a group: those that are easily cognizable without a consideration of the surrounding objects (for example, color), and those that are derived from comparison of the intended referent with the surrounding objects (for example, size). It is only the latter that is derived from an incremental process.

Imposing a separation between the two properties provides a way to neatly account for the kinds of redundant or "unnecessary" information that tends to be included in reference. Those properties that are easily cognizable without consideration of the surrounding objects are generated regardless of whether they rule out other members of the contrast set. Those properties that are derived from comparison processes are generated by modelling the comparison process algorithmically; by incrementally parsing each item in the contrast set. This model generates the kinds of referring expressions observed in natural language: uniquely identifying, underspecified, and overspecified expressions. I now discuss the implementation of the incremental process.

Incrementally Ruling out the Contrast Set

Incremental referring expression generation algorithms proceed by selecting properties that rule out other competing referents. The entire contrast set is available at the time when referring expression generation begins. Only attribute-value pair selection is incremental,

manifested from incrementally ruling out members of the entire contrast set. This implementation misconstrues Pechmann’s findings. Humans begin to generate reference *before* they are done scanning the entire scene. At the time of reference, they do not have access to the entire contrast set, but rather, *examination of the contrast set is incremental*.

The algorithm I propose follows this more literal interpretation of Pechmann’s work. The algorithm utilizes as input a series of vectors, which correspond to the members of the context set. The vectors of the contrast set are analyzed incrementally, following a prespecified order to generate modifiers such as *tall* and *wide*. The vector of the intended referent is analyzed separately, to generate modifiers such as *red* and *velvet*. This approach successfully generates natural referring expressions.

1.2.3 Summary

Previous approaches to generating natural sounding referring expressions have justified uniqueness on the basis of Grice (and therefore also suggested that such expressions would be “natural”), which is an assumption I reject. Tests on communication between people have shown two ideas that are integral to generating natural sounding referring expressions:

1. Humans generate referring expressions that do not uniquely identify referents
2. Humans generate referring expressions with descriptors that are not distinguishing

To formalize this in a computational model, I rely on corpus studies and psychological work, particularly focusing on the findings from Pechmann (1989). Following Pechmann, I impose a separation of two basic kinds of properties: those that are easily cognizable without a consideration of the surrounding objects (for example, color), and those that are derived from comparison of the intended referent with the surrounding objects (for example, size). An incremental analysis of the contrast set is used to derive only the latter properties. This analysis differs greatly from previous work to date, which has not prioritized naturalness but has implemented a different interpretation of Pechmann’s findings.

1.3 The Approach

The previous two sections have outlined basic ideas on how to generate natural reference. Using these ideas, I construct an algorithm that generates natural referring expressions. I now outline the algorithm, and define specifically the kind of reference it generates.

As mentioned in the Section 1.1, approaches to the generation of referring expressions largely focus on creating distinguishing descriptions. However, a distinguishing description is by definition not a natural form of reference, and so cannot be used for the current work. Instead, I isolate two types of reference, which both take the form of a noun phrase with prenominal modifiers. I call these types *distinguishing reference* and *describing reference*. I define the term **distinguishing reference** to be reference aimed at picking out an entity from a group of entities, and the term **describing reference** to be reference aimed at conveying information about a referent, regardless of whether that information provides unique identification (or is actually true of the referent). This separation follows previous research on reference (Donnellan, 1966; Kronfeld, 1989). Research on human referring expression generation, without imposing such a separation, generally tends to examine expressions that uniquely identify an object from a group of objects. This research may be applied specifically to what I define as distinguishing reference.

To consider the difference between the two forms of reference, imagine the case of discussing art from around the world. In this situation, one may mention a *Byzantine tapestry*, or a *helpful curator*. These expressions do not necessarily pick out one item from a group of items, but rather, introduce a referent along with descriptors relevant to the conversation. To pick out that same tapestry when presented with other kinds of art, one may tend to identify the *brown tapestry* or *the tapestry with a picture in it*. This speaks to a basic point: Picking out an item from a group of items is a different task than naming an item in a summary or a discussion, where the goals of the act as a whole determine which descriptors are used to identify the referent.

The two forms require different generation approaches. To generate distinguishing reference necessitates considering what properties of an entity will identify it; to generate de-

scribing reference draws from the desire to convey information about an entity, or describe it in some way that is relevant to the context of the communicative act. The algorithm developed in this thesis generates distinguishing reference.

To generate distinguishing reference, fundamental properties of the entity being referred to must be analyzed. This is a content determination task, and represented in my algorithm as a text planning function. This function returns a variety of descriptors. The second stage of this algorithm is a microplanning function, which can select specific prenominal modifiers for natural reference based on the linguistic personality desired of the system. The ordering of the modifiers that are not derived incrementally are determined by the algorithm's surface realization function using a modifier type system developed in Chapter 4. This modifier type system may be used to generate any noun phrase with prenominal modifiers. The algorithm as a whole is tested in Chapter 5.

A flowchart illustrating how my algorithm for distinguishing reference fits within a natural language generation system is shown in Figure 1.1. The boxes on top represent the components of the algorithm developed in this thesis, and the boxes below represent where they may be placed in an NLG system. Chapter 3 introduces the text planning and microplanning components of the algorithm to generate distinguishing reference. Chapter 4 introduces the surface realization component, and a modifier type system is developed that may be used to generate both distinguishing reference and describing reference.

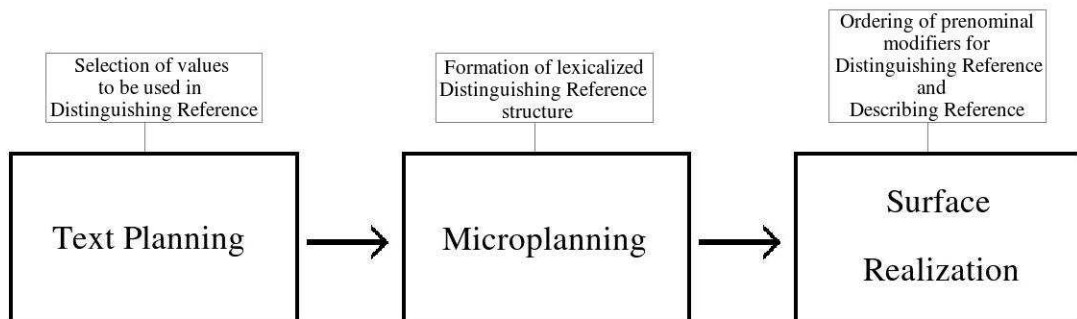


Figure 1.1: Distinguishing Reference and Describing Reference

1.3.1 Distinguishing Reference

I may now bring the ideas outlined in this chapter to a head: For an algorithm to generate a form of natural reference, distinguishing reference, some modifiers must be generated based on the fact that they convey salient properties of the referent and other modifiers must be generated based on an incremental analysis of the relations that hold between the intended referent and the contrast set. The algorithm introduced in Chapter 3 does this by separating out two main kinds of properties. I now define these to be **absolute properties**, those properties that are easily cognizable without a consideration of the surrounding objects (for example, color), and **gradable properties**, those that are derived from comparison of the intended referent with the surrounding objects (for example, size).

These properties can be used to produce several referring expressions for each referent. Gradable properties are parsed by the algorithm by an incremental analysis of the contrast set, while absolute properties are stored and then ordered in the surface realization stage by using a modifier type system. As discussed earlier, the absolute properties are not chosen by comparison with any items in the contrast set, and so may include information that is not uniquely distinguishing. The gradable properties selected to identify the referent are derived from incrementally comparing the intended referent to the items in the contrast set. The algorithm derives modifiers from these properties before passing them in a structure to the surface realization component.

The output of this algorithm, once absolute properties are ordered in the surface realization stage, is a noun phrase with prenominal modifiers. Each referring expression is structured so that absolute properties are closer to the head noun than gradable properties. Some examples of the phrases the algorithm generates are given in Figure 1.2.

1.3.2 Surface Realization

To order the modifiers derived from absolute properties, the surface realization component of the algorithm utilizes a modifier type system. The type system includes adjectives (*small*

- (1) green box
- (2) large green box
- (3) large green velvet box

Figure 1.2: Example Distinguishing Reference

cat), nouns that modify other nouns (*metal cat*), and modifying verb forms (*tired dancing cat*). Ordering based on these types is straightforward, and outlined in Chapter 4. This modifier type system is developed for the algorithm, but may be used to order the prenominal modifiers of any noun phrase.

The modifier type system is a separate component, used by the algorithm. It may be provided by the system, and in Chapter 4 I break from a discussion of the algorithm in order to develop the modifier type system. This is followed by an evaluation, which shows the system to be extremely effective at ordering modifiers prenominally. I will summarize the system briefly here.

The type system is based on the distribution of prenominal modifiers in natural language, and is developed using the Google N-Gram corpus (Brants and Franz, 2006) and WordNet (Miller, 2006). A natural maximum of four modifiers preceding the head noun is assumed, and from this assumption, nine modifier classes are derived. These classes are used to order modifiers prenominally, and provide a way to generate orderings of modifiers that have strict positional preferences (*big red hat* and not *?red big hat*) as well as orderings of modifiers that have more loose positional preferences (*domestic organic beer* and *organic domestic beer*).

Instead of requiring an ordering of individual modifiers² along with each referring expression to be generated, as previous research has proposed, this system allows modifiers to belong to larger classes, where the classes indicate the ordering prenominally, and the surface realization component simply needs to order the adjectives based on which classes they belong to. An unordered grouping of prenominal modifiers, along with the head noun,

²More precisely, instead of requiring a prespecified ordering of attribute-value pairs.

are therefore required as input to the surface realization component. That is, the process of ordering pronominal modifiers to identify a referent requires a structure for which the task of content determination has already been completed. The approach to surface realization can therefore be used to generate any form of reference where a head noun is preceded by modifiers.

1.3.3 Evaluation

To evaluate the effectiveness of the algorithm, Amazon's MechanicalTurk program (Amazon, 2008) is used. Expressions are elicited from human participants, and I analyze whether each referring expression can be generated by the algorithm I introduce as well as by the Incremental Algorithm. I find that my algorithm is capable of generating the vast majority of observed human referring expressions, and more successful than the Incremental Algorithm.

1.4 Thesis Organization

This thesis is presented in six chapters. The following chapter outlines the field of Referring Expression Generation, documenting the literature that has been most influential in this domain. Chapter 3 introduces the algorithm for generating referring expressions for distinguishing reference, and discusses its implementation. Chapter 4 presents an approach to ordering pronominal modifiers, which can be used for both distinguishing reference and describing reference. Chapter 5 examines the effectiveness of the proposed algorithm by presenting an evaluation of how many expressions elicited from humans in a visual presentation task can be predicted by the algorithm. Finally, Chapter 6 summarizes the ideas and findings from this thesis, and offers a direction for future research.

Chapter 2

LITERATURE REVIEW

To approach an analysis of the generation of referring expressions, I will first present a discussion of the papers that have informed this field. In this chapter, I trace the history of research in referring expression generation and bring the reader to an understanding of where the field currently stands. The areas I discuss are drawn from work in philosophy, computer science, and psychology. The papers discussed are those most significant to the current work, but the brief summaries I provide here are a very small piece of the breadth of knowledge conferred by these authors.

This chapter also provides a short introduction to approaches that have been taken to classify adjectives in order to determine their position prenominal. I offer a different kind of prenominal modifier classification scheme in Chapter 4, for use in a Surface Realization component of generation systems. This classification scheme is the last piece of the algorithm introduced in this thesis, and the papers outlined here present the background from which my own approach is developed.

2.1 The Beginning

In 1967, H. P. Grice gave a series of lectures at Harvard on the nature of conversation. Grice was a philosopher and a linguist, and these lectures were a product of years of research on the relationship between logic and natural language. Grice sought to represent conversation and natural language in logical terms, and believed that the fundamental processes underlying communication could be represented in logical form. Notes from these lectures circulated for years, and in 1975 a journal published a portion of this work in an article called “Logic and Conversation” (Grice, 1975). This article outlined one of the basic ideas Grice had developed, the idea that conversation operates by obeying to four basic maxims. These maxims have come to be known as the Gricean maxims, and they are listed in Figure 2.1.

Quantity

1. Make your contribution as informative as is required.
2. Do not make your contribution more informative than is required.

Quality – Try to make your contribution one that is true.

1. Do not say what you believe to be false.
2. Do not say that for which you lack adequate evidence.

Relation – Be relevant.**Manner** – Be perspicuous.

1. Avoid obscurity of expression.
2. Avoid ambiguity.
3. Be brief (avoid unnecessary prolixity).
4. Be orderly.

Figure 2.1: The Gricean Maxims

These maxims became extremely influential. Grice's work related natural language to logical predications, a realm where clearly defined rules can operate. With this in place, language could be seen even more clearly as deriving from interacting principles, where different kinds of utterances are a product of consistent and predictable forms. That is, Grice helped pave the way for language to be seen as fundamentally logical.

This approach to language had a lasting impact on NLG, among other fields. Grice's maxims are well-suited to the intrinsically logical processes of computers, and have provided the basis for a great deal of work utilizing computers to produce natural sounding utterances.

However, these maxims are not aimed at capturing natural language. Grice writes,

The conversational maxims ... are specially connected (I hope) with the particular purposes that talk (and so, talk exchange) is adapted to serve and is primarily employed.

(Grice, 1975: 58)

The nature of this connection is not specified, but the article is clearly not aimed at capturing the way people actually speak. Instead, it provides guidelines for how people should speak if they want to communicate with optimal effectiveness. It does not follow that the Gricean maxims can be used to guide naturalness.

The discrepancy between natural language and language obeying the Gricean Maxims is pointed out by Grice himself, in the same article where the maxims are introduced. He writes:

I have stated my maxims as if this purpose were a maximally effective exchange of information; this specification is, of course, too narrow.
(Grice, 1975: 58)

In this thesis, I follow this advice.

Around this same time, Terry Winograd was helping to create the beginnings of the very field that would rely on Grice’s intuitions. Working in the M.I.T. Artificial Intelligence Laboratory, Winograd spent the late sixties and early seventies developing a program called SHRDLU, one of the first attempts to understand language by using a computer. The program, which was eventually documented in Winograd (1972), displayed a variety of blocks and pyramids. Users could enter instructions for the computer to “pick up” or “find” the different shapes, and the computer would “understand” the instructions, or else ask for clarification.

Creating the ability for computers to understand language in this way, to interact with and respond to a user, gave rise to an entire area of research. This has come to be known as Natural Language Processing, comprised of the two subfields of Natural Language Understanding and Natural Language Generation. It is in the field of Natural Language Generation that the current work proceeds.

2.2 The Field of Natural Language Generation Begins

By the beginning of the 1980s, work in Natural Language Generation had exploded. David McDonald (1980) wrote a thesis examining the utilization of computers to generate language

using constraint-based processes, while Douglas Appelt (1981) worked towards analyzing how to capture the different goals at play when communicating. This research helped pave the way for work examining how computers could generate and parse linguistic phenomena.

In 1985, Appelt began to address specific issues in language generation. In a discussion of the problem of creating natural sounding reference to entities, Appelt (1985) introduced the problem of planning **referring expressions**, expressions that identify an entity. Appelt utilized the KAMP system to approximate human production of sentences using first order logic, and stressed the difficulty in capturing how humans refer. He noted that there are many types of object reference, satisfying multiple goals. Appelt saw that producing natural utterances requires a powerful system, capable of reasoning not only about the physical world, but beliefs and intention.

Through the rest of the eighties, Kathleen McKeown (McKeown, 1985), Michael Zock (Zock et al., 1986), and others further explored the intricacies of natural language generation. Formally explicit algorithms for the generation of natural language began to appear. And in 1989, the first algorithm for the generation of referring expressions was introduced.

2.3 Referring Expression Generation

In 1989, Thomas Pechmann published a paper illustrating that reference to objects is often produced incrementally (Pechmann, 1989). As discussed in Chapter 1, this is the process of **incremental speech production**. Using eye-tracking techniques, Pechmann presented subjects with a variety of objects, and asked them to name one. This exercise showed that people begin producing descriptions of items before they scan the entire scene, and in fact, describe the object as they fixate on the other objects in the scene. Pechmann argues that this suggests that reference to objects is an incremental process, where the features necessary to distinguish an object are not formulated before the utterance begins, but rather chosen as the utterance progresses.

Further findings from Pechmann's study include the fact that speakers produce reference that is **overspecified**, with descriptors that are true of all objects in a scene and not necessary for unique identification. Pechmann also showed that, in his tasks, people include

color descriptors in 98% of their utterances to pick out objects.

This research has become the most influential psycholinguistic work in Referring Expression Generation. For the first time, the cognitive processes involved in reference began to be made clear. This has made it possible to formulate algorithms that parse scenes similarly to how humans do, and so generate natural sounding reference.

However, there are elements of Pechmann's work that are often overlooked. To describe incremental speech production, Pechmann writes:

...The speaker initially pays attention to the target object without seriously considering the context...The speaker starts to articulate features of the target object which are easily cognizable. One such feature is, for instance, color, which can be determined without considering any other contextual objects. In contrast, describing an object as either 'small' or 'large' required comparison processes...Such an incremental strategy of object naming implies that the speaker does not absolutely intend to mention only distinguishing features of the target object while carefully trying to avoid the incorporation of any non-distinguishing information into his utterance. **It is rather characteristic of such a strategy that the speaker articulates features of the target before he had determined whether they are distinguishing or not.**

(Pechmann, 1989: 98. Boldface my own).

Pechmann proposes that there are actually two distinct kinds of features at play in generating reference, and incremental speech production proceeds for only one; this is comprised of those features that require comparison processes. The nuances of incremental speech production are discussed in detail in Chapter 1.

Later that year, Robert Dale introduced the first explicit algorithm for the generation of referring expressions (Dale, 1989). In this paper, Dale describes the referring expression generation mechanisms used in the system EPICURE, and introduces what came to be known as the Full Brevity Algorithm. This algorithm produces the minimal description of an object necessary to uniquely identify it. That is, this algorithm produces expressions with the

fewest amount of descriptors necessary to distinguish an object in a group of objects.

This kind of reference is defined by Dale to be a **distinguishing description**. This term is adopted in all later work in the area. The definition Dale uses is as follows:

Suppose that we have a set of entities U such that

$$U = \{x_1, x_2, \dots, x_n\}$$

and that we wish to distinguish one of these entities, x_i , from all the others.

Suppose, also, that the domain includes a number of attributes (a_1, a_2 , and so on), and that each attribute has a number of permissible values (v_{n1}, v_{n2} , and so on); and that each entity is described by a set of attribute-value pairs. In order to distinguish x_i from the other entities in U , we need to find some set of attribute-value pairs which are together true of x_i , but of no other entity in U . This set of attribute-value pairs constitutes a *distinguishing description* of x_i with respect to the context U .

(Dale, 1989: 71)

In Dale's terminology, the object being referred to is the **intended referent**, and the group of entities that is not the intended referent is the **contrast set**. These are defined again for my own approach in Chapter 3.

Two years later, Ehud Reiter published a paper addressing some of the limitations of the Full Brevity approach to referring expression generation. In this paper, Reiter proposes a new version, called the Local Brevity Algorithm. This algorithm avoids some of the computational complexity of the first, checking that each description component cannot be replaced by a briefer description component without losing discriminatory power.

Later that year, Dale and Haddock (1991) introduced a procedure that could generate referring expressions involving relations, using a depth-first search. Problems arise from this algorithm when it gets stuck recursively trying to identify the referent and the object used to identify the referent, generating descriptions such as "the cup on the floor which is holding the cup which is on the floor which is..."

Work in the generation of referring expressions began to focus specifically on naturalness with the publication of the paper “Computational Interpretations of the Gricean maxims in the Generation of Referring Expressions” (Dale and Reiter, 1995). This paper underlies much of the following work in the generation of referring expressions. In this paper, Dale and Reiter introduce the Incremental Algorithm for generating distinguishing descriptions of referents.

Drawing on the Gricean maxims and the idea of incremental speech production developed by Pechmann, the Incremental Algorithm aims to pick out a referent by incrementally analyzing the properties that are true of the referent, and finishing when the intended referent has been uniquely identified.

As in previous algorithms, the Incremental Algorithm operates on the properties as represented in terms of attribute-value pairs. It proceeds by iterating through attributes in a predefined order. For each attribute, it checks whether specifying a value for that attribute would rule out at least one referent in the current discourse that has not already been ruled out. If it does, that attribute is selected. The algorithm then chooses a value for that attribute that is known to the user and that is as basic as possible while ruling out the maximum number of referents possible. Once this descriptor is selected, the algorithm adds it to a set to be used in the referring expression. This continues until there are no longer any referents confusable with the intended referent. Dale and Reiter define the decision on which set of properties to single out the target object as the **content determination** problem. Pseudo-code for this algorithm is given in Figure 2.2.

One interesting piece of this algorithm is the UserKnows function. This was not developed in the paper, but it provides a way to account for the interlocutor’s model of the common ground. The algorithm thus has a way of reasoning about shared knowledge. The algorithm also calls to a BasicLevelValue function and a MoreSpecificValue function, which are not defined as part of the algorithm but provide system-dependent information on the basic and more specific forms for each selected attribute.

Dale and Reiter also overview the previous algorithms in the field, and argue that the Incremental Algorithm is preferable, as it is less computationally complex and more reflec-

MakeReferringExpression(r, C, P)

```

L ← {}
for each member Ai of list P do
  V = FindBestValue(r, Ai, BasicLevelValue(r, Ai))
  if RulesOut(<Ai, V>) ≠ nil then
    L ← L ∪ {<Ai, V>}
    C ← C - RulesOut(<Ai, V>)
  end if
  if C = {} then
    if <type, X> ∈ L for some X then
      return L
    else
      return L ∪ {<type, BasicLevelValue(r, type)>}
    end if
  end if
end for
return failure

```

FindBestValue(r, A, initial-value)

```

if UserKnows(r, <A, initial-value>) = true then
  value ← initial-value
else
  value ← no-value
end if
if (more-specific-value ← MoreSpecificValue(r, A, value)) ≠ nil ∧
(new-value ← FindBestValue(A, more-specific-value)) ≠ nil ∧
(|RulesOut(<A, new-value>)| > |RulesOut(<A, value>)|) then
  value ← new-value
end if
return value

```

RulesOut(<A, V>)

```

if V = no-value then
  return nil
else
  return {x : x ∈ C ∧ UserKnows(x, <A, V>) = false}
end if

```

(Dale and Reiter, 1995: 257)

Figure 2.2: The Incremental Algorithm

tive of what humans actually do. They also state that “[conclusions from the paper], in particular that computationally simple interpretations of the Gricean maxims of conversational implicature should be used, will also prove applicable to other referring expression generation tasks” (234). These ideas have largely been adopted in the field.

2.4 What Psychological Evidence Has to Say

Other than the seminal work by Pechmann, there is some psychological work that has specifically addressed different facets of how people refer to objects. These studies investigate what properties of objects people use when trying to pick out a single object, and how reference and dialogue proceeds. There is not a great amount of literature on the topic, but any algorithm that attempts to mimic human production is well-advised to draw from what is available.

In 1985, Clark and Wilkes-Gibbs organized a study using tangram figures (Clark and Wilkes Gibbs, 1986), which illustrated that in a cooperative identification task, speakers use long, descriptive noun phrases to first identify a referent, and these phrases become shorter over time. They introduce the idea of a **conversational model**, where people communicate and act collaboratively, establishing meaning moment-by-moment. This analysis differed from the prevailing view that establishing reference is largely autonomous, where the speaker has complete control and responsibility over referent determination, and the listener is a passive understander.

Landau and Jackendoff (1993) added to the psychological research on object reference by analyzing the geometric properties that underlie object nouns. The authors examine which properties are utilized for distinguishing one object from another, and posit a few key components of object reference. The idea most integral to their analysis of reference is that intersecting axes define an object’s relation to space. These are outlined as follows:

Generating axis: This is an object’s principal axis, and can be seen as running through the top and bottom of the object.

Orienting axes: These are secondary and orthogonal to the generating axis and to each

other (corresponding to the front/back and side/side axes).

Directed axes: These differentiate between the two ends of each axis, marking top/bottom and front/back.

(Landau and Jackendoff, 1993: 221)

With these axes in place, the authors make a distinction between surface-type and volume-type objects, suggesting that adjectives are used differently depending on which category an object falls into. **Surface-type** objects are those that principally extend in two dimensions (such as a record), while **volume-type** objects are those that extend in all three (such as a box). The utilization of these two types to describe objects can be seen in the fact that, for example, a record only needs to be wide to be called a big record (not thick), while a box needs to be both wide and tall to be called a big cube (otherwise it would just be called tall or wide).

Appropriate reference to parts of these objects can be seen as stemming from their axial structure. For example, if an object is long and narrow, it has a horizontal generating axis that is longer than the other axes, and can thus be said to have ends; the regions at the termination of the axis. This idea provides a way to represent the human perception of objects, and the generation of referring expressions may benefit from incorporating these ideas of object properties.

2.5 Improvements Since the Introduction of the Incremental Algorithm

Since the publication of the Dale and Reiter (1995) paper, many steps have been made towards advancing the Incremental Algorithm's scope. Approaches to the generation of referring expressions have used the Incremental Algorithm to build reference to sets, generate more complex kinds of modifiers, and expand linguistic naturalness.

Horacek (1997) attempted to capture more elements of human reference. This algorithm introduced in this paper provides for situations where the Incremental Algorithm generates unnatural utterances. The algorithm imposes a complexity limit, such that ambiguous (not fully distinguishing) referring expressions are produced once the referring expression has

reached a maximum number of descriptors. The algorithm is also extended to handle situations where no uniquely identifying expression can be reached, and does not add descriptors that are inferable from the descriptors already chosen for a given distinguishing description. Additionally, the algorithm rejects descriptors if they cannot be lexicalized with the given linguistic resources.

A few years later, Stone (2000) extended the Incremental Algorithm for the generation of reference to sets. This approach mirrors that taken to refer to singular entities, but constructs the search space to apply to groups of items.

van Deemter (2002) extended the Incremental Algorithm to allow for descriptions that involve more than just an intersective combination of properties. He explains that the Incremental Algorithm does not handle cases where some attribute-value pairs holding true of an entity are overlapping (for example, a ball may be both red and orange), and proposes an approach where the algorithm checks if each attribute chosen with a particular value has another value which identifies the same referent. If this is the case, the algorithm checks if it is a subset of the first. If it is not, it includes the other value as well. This paper also examines reference to sets, and using disjunctions as well as conjunctions of properties. The descriptions his new algorithm generates are distinguishing descriptions using both conjunctions of positive properties and negative disjunctions of properties.

Around this same time, Krahmer et al. (2003) developed a new kind of algorithm for the generation of referring expressions. This algorithm is graph-based, which is in stark contrast to most other work that attempts to refine the Incremental Algorithm without fundamentally changing it. In their paper, the authors formalize a scene (consisting of a set of objects with various properties and relations) as a labeled directed graph and describe content selection as a subgraph construction problem. Using a graph-based approach allows for better generation of **relational expressions**, referring expressions that include references to other objects. This is possible because both properties and relations are formalized in the same way: as edges in a graph.

Krahmer et al. construct referring expressions based on the kind of graph that can be placed over a larger graph of the knowledge base. In this structure, arcs between referents

correspond to relations, such as *next to* and *left of*, and concentric circles represent arcs that stem from and return to the same referent, representing descriptors of that referent. Weights for each circle dictate the order in which descriptors are chosen, where those with the least weight are chosen first. In this way, distinguishing descriptions for referents can be created by following a path to the referent.

Also in this period, Claire Gardent worked on a variety of approaches for the generation of referring expressions, creating algorithms that generate maximally brief distinguishing descriptions (Gardent, 2002), that do not rely on incremental speech production, and that use a discourse model to derive the kinds of referring expressions that are used throughout discourse (Gardent et al., 2004).

van Deemter (2004) continued work on extending the abilities of the Incremental Algorithm. This was a continuation of earlier research on vagueness (van Deemter, 2000), and utilized some of Krahmer et al.’s findings. Specifically, the difference between **absolute properties**, properties that are inherent to the noun, and **gradable properties**, properties defined by the contrast set, was addressed. This paper looked at just generating gradable properties, creating a way for the Incremental Algorithm to generate pronominal modifiers that convey gradable information, such as *big* and *thin*. This approach assumes absolute measurement values for the intended referent, and derives relative terms based on the contrast set.

Areces et al. (2008) present a Description Logic approach to the generation of referring expressions. This approach is faster than any algorithm to date, subdividing the items in the context set into groups until a unique description for the intended referent is created. The drawback to this approach is that it requires a post-processing step to convert the sometimes multiple disjunctions into expressions that are more natural. It also does not always uniquely identify referents, which is true of natural language, but it has not been shown that the occasional ambiguity this algorithm produces reflects the referring expressions generated by people.

2.6 Prenominal Ordering of Adjectives

Referring Expression Generation algorithms often use a pre-specified listing (or weighting) of referent properties, which are realized as modifiers. The approach I take in this thesis does not use any such listing of properties, but instead utilizes a modifier type system. This is introduced in Chapter 4. I will now turn to a brief discussion of previous approaches at predicting prenominal adjective ordering based on adjective classification. Although much work has looked specifically at adjectives, the results from these studies can likely be generalized to other prenominal modifiers (nouns and verbs) as well.

It is often noted in studies on prenominal adjective ordering that the placement of the adjectives in phrases such as *the large red thing* is preferable to the placement of the adjectives in the phrase *the red large thing*. Many theories have been developed to account for this kind of phenomenon, with most proposing a relationship between the semantics of an adjective and its position prenominally. Modelling adjective ordering in noun phrases is an integral part of generating natural referring expressions, and determining the factors that affect adjective ordering is a problem that has persisted for millennia. Some approaches date as far back as 350 BC (Panini, 350 BC, as cited in Cooper and Ross, 1975).

Whorf (1945) distinguishes two groups of adjectives, corresponding to **inherent** and **non-inherent** qualities of the noun they modify. Whorf’s proposed ordering based on these categories has non-inherent adjectives preceding inherent adjectives, thus *large red house* is grammatical, exhibiting a common ordering of prenominal adjectives. Whorf points out that the order may be reversed to make a balanced contrast, but only by changing the normal stress pattern, and the form is “at once sensed as being reversed and peculiar” (Whorf, 1945: 5).

A similar approach to predicting the ordering of adjectives based on the semantic relationship between adjective and head noun is discussed by Ziff (1960, as cited in Richards, 1975 and Wulff, 2003). Ziff explains that adjectives that are semantically close to the head nouns they modify are tied to these nouns and so are selected with them (e.g., *bright light*, *playful child*). Adjectives that are not tied to head nouns in this way will appear with a

greater range of nouns (e.g., *good*, *pretty*), and so appear farther from the head noun. It follows that adjectives that appear farther from the head noun are those that are applied to a wider class of nouns, and have less semantic similarity to the head noun.

Vendler (1968) presented the natural order of adjectives as a function of transformational operations applying to adjective classes. His claim is that the order of the application of the transformational rules affects the order of adjectives prenominal. Classes of adjectives in this view are grouped by their relationship to a proposed underlying verb. This is characteristic of a transformational-based approach: Adjectives are seen as existing first in predicate structures, where the sentence transforms into a nominalization that includes an ordering of the adjectives.

Danks and Glucksberg (1971) concluded that prenominal adjective ordering is “determined by the pragmatic demands of the communication situation” (Danks and Glucksberg, 1971: 66). This is called the **pragmatic communication rule**. The idea developed by the authors is that adjectives that are more discriminating precede less discriminating ones, but the specific ordering of adjectives is dependent on the context.

It is therefore common to try to predict the ordering of prenominal adjectives by proposing some relationship between the surface order and an underlying factor, usually semantics but also syntactic transformational processes. Most work exploring prenominal adjective order focuses on adjective semantics, proposing that adjectives are ordered according to a scale. The exact nature of the scale differs, but research tends to favor an analysis where adjectives closer to the head noun have a stronger association with it than adjectives that are farther away. As will be discussed in Chapter 4, my approach to determining the ordering of prenominal modifiers backs away from proposing a relationship between modifier position and any underlying factors, but does not preclude such interpretations.

2.7 Summary

Since 1989, many algorithms have been developed for the task of generating referring expressions. The algorithms differ in their approach, focusing on computational efficiency, unique identification, and psycholinguistic realism. Since 1995, generating natural reference has

been a particularly active topic. Research on generating referring expressions has largely utilized the Incremental Algorithm as a base, expanding its scope to handle a variety of reference techniques.

Although research has focused on expansion to different kinds of linguistic phenomena, very little work has been done on naturalness as such. Approaches for natural generation do not diverge from the core tenets of the Incremental Algorithm. The nature of the adjectives chosen for generation is usually treated as system-dependent, with no definitive work leading the way as to what these adjectives are, or how they will be ordered or weighted. Further, the interplay between adjective generation and contrast set analysis is an area that has yet to be fully explored.

Exploration of the nature of human reference may result in different approaches for generating referring expressions. These may fundamentally differ from the Incremental Algorithm, aiming specifically at naturalness and not necessarily unique identification of intended referents. Herein lies my own approach, and hopefully, the beginnings of research to come.

Chapter 3

THE ALGORITHM**3.1 Overview**

This chapter introduces an algorithm for the generation of **distinguishing reference**. Distinguishing reference is defined to be reference to pick out an entity from a group of other entities. As discussed in Chapter 1 and Chapter 2, the **intended referent** is defined to be the entity to be picked out, the **context set** is defined to be the group in which it exists, and the **contrast set** is defined to be the group of entities not including the intended referent. The contrast set can be composed of 0 or more entities; that is, the intended referent in distinguishing reference may not have any other competing entities from which to be distinguished.

The algorithm contains a content determination component, which may be implemented in a Text Planner; a lexicalization component, which may be implemented in a Microplanner; and a realization component, which may be implemented in a Surface Realizer. (See Section 1.1 for an overview of these components.) The content determination component is the core of the algorithm. It creates a basic structure for generating distinguishing reference by storing the properties true of the intended referent and analyzing the contrast set to derive size descriptors. The microplanning component selects which descriptors in particular will be used for the final expression. The surface realization component can then generate distinguishing reference by ordering the modifiers selected to convey absolute properties. A general flowchart of this algorithm is presented in Figure 3.1.

3.2 Main Ideas

The algorithm introduced in this chapter generates distinguishing reference for objects in a visual presentation task. The specificity of this approach is intentional: If one kind of

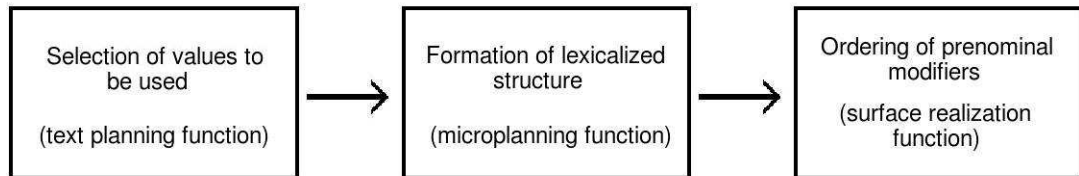


Figure 3.1: Flow Chart of Algorithm for Distinguishing Reference

distinguishing reference can be generated by obeying the basic tenets of this algorithm, then it is a reasonable starting point for generating other kinds of distinguishing reference as well. This is also motivated by the data available. Most studies on human-produced referring expressions examine reference to objects presented visually, and drawing conclusions for other kinds of reference requires further investigation. An evaluation of the algorithm is presented in Chapter 5.

As discussed in Chapter 1, the goal of this algorithm is to generate natural sounding reference. The ideas implemented to achieve this goal are:

1. Incremental examination of the contrast set
2. Descriptors split into those inherent to the intended referent and those derived from the contrast set
3. Ordering of descriptors based on the order of the contrast set and a surface realization component

Also as discussed in Chapter 1, it has been noted by those working on referring expression generation that there is a fundamental difference between **absolute properties** (e.g., color) and **gradable properties** (e.g., size) (Dale and Reiter, 1995; Krahmer et al., 2003; van Deemter, 2004). The algorithm here separates these two kinds of properties and analyzes them in different ways. This helps form natural sounding reference.

Absolute properties and gradable properties make up two separate inputs. The absolute properties are converted directly to modifiers for generation. The gradable properties are

computed incrementally for each item in the contrast set. It is assumed that the items in the contrast set will be ordered by salience, where the first items to be ruled out are the first items that are likely to be fixated on by a human in a presentation task.

3.3 *Towards Generation*

Before introducing the algorithm, I will discuss its placement in a Natural Language Generation system. (See Table 1.1 for a general outline of NLG systems.) Two main functions from this algorithm create a structure for distinguishing reference. This structure is to be passed to a Surface Realizer, which completes the algorithm. The Surface Realization step is discussed in Chapter 4. In this chapter, the first two main functions of the algorithm are discussed. The first solves what properties will be converted to prenominal modifiers. This is the text planning section of the algorithm. The second function picks out precisely which modifiers will be used, where several different expressions can be created depending on the linguistic personality desired. This is the microplanning section of the algorithm. The output of the microplanning stage provides the surface realization component of the algorithm with a list of ordered modifiers denoting gradable properties, followed by a list of unordered modifiers denoting absolute properties, followed by the head noun. These unordered modifiers are then ordered in a surface realization component by using the system's modifier type system.

This organization is different from the traditional approach to referring expression generation. Most referring expression generation algorithms do not impose clear delineations between the stages of Text Planning, Microplanning, and Surface Realization. Algorithms for generating referring expressions are generally treated as microplanning algorithms, although they determine content as well as generate. This is likely because the final surface string is so closely tied to the input properties. It is also due to the evidence that humans generate referring expressions incrementally, meaning they do in fact determine content as they generate.

The reason that this algorithm does not follow in this tradition is simple: Natural language generation systems generally do not traverse the path from text planning to surface

realization for every word or even every phrase. Instead, generation systems generally analyze information in larger chunks, first text planning an entire sentence (or more), then moving to microplanning, and then moving to surface realization. That is, natural language generation systems generally analyze information that is generated as a portion of cohesive text, instead of analyzing information phrase-by-phrase. This puts previous approaches to referring expression generation in an odd place, and many have noted that despite the usual flow of NLG systems, referring expression generation algorithms are a special case, performing *content determination* – usually a task that takes place in the Text Planner – as well as creating a structure that may be generated immediately by a Surface Realizer.¹

The approach here actually allows the generation of referring expressions to be split between the three tasks, following the flow of natural language generation. If a system were constructed where each item chosen in the Text Planner immediately went through to the Surface Realizer, comparable to human production, *this approach could still generate natural reference*. The algorithm would have to be rearranged, but the basic ideas underlying it still give rise to natural reference. In fact, changing the algorithm in this way would allow it to perform even better. This is because gradable properties are ordered as an effect of content determination, generally part of the Text Planning stage. Modifiers that denote absolute properties, on the other hand, make up an entire group that are ordered in the Surface Realization stage. This separation ensures that gradable properties always come first in generation, followed by absolute properties. But the only reason that the two kinds of properties are grouped together in the output of the text planning stage of my algorithm (and subsequently in the microplanning stage) is to allow language generation to occur in larger chunks. Preserving the separation between these two kinds of properties throughout the algorithm would not be necessary if surface realization could proceed immediately after deriving the first gradable property descriptor.

¹It deserves note that referring expressions are not the only things that are generated incrementally, and treating them as a special case outside of the general flow of NLG may not be merited. In fact, Pechmann proposes that all utterances proceed incrementally. He focuses in particular on referring expressions, but later work supports the idea that entire sentences proceed incrementally as well (cf. Griffin and Bock, 2000).

3.4 Overview of the Algorithm

As has been noted, studies on human reference illustrate that people produce descriptions of objects incrementally, first fixating on the object and then naming it as they fixate on the objects that make up the surrounding context (Pechmann, 1989; Griffin and Bock, 2000). It follows that the properties inherent to the object being named are considered as one chunk. These are the absolute properties.

The other properties that are considered are those that are defined by the contrast set. Following van Deemter (2000, 2004), I call these gradable properties. These are considered in the order supplied by the system. This order should correspond to visual salience,² but in the current approach, it corresponds to proximity to the intended referent. In this way, gradable properties are built up by incrementally analyzing each of the members of the contrast set.

This approach fits in neatly with the approach suggested by Pechmann, who pointed out that “...a red pencil is as red as a red car given the same redness, but a small hippopotamus is already quite big compared to a big mouse” (99). The idea Pechmann presents is that adjectives which are “more definite” in their meaning come closer to the noun than adjectives that are not – as such, size precedes color. This is the **definiteness of denotation** principle, proposed by Martin (1969a), and similar ideas have also been proposed by Whorf (1945), Ziff (1960), and Danks and Glucksberg (1971) (see Chapter 2 for a review of this material).

The algorithm therefore utilizes two general modifier types: those that are defined by the intended referent (absolute properties) and those that are defined by the contrast set (gradable properties). With this in place, absolute properties are analyzed before gradable properties. Once analysis of the gradable properties begins, the ordering of the final output begins. This creates a structure where the “definite” or absolute modifiers come closer to the head noun than those that are not.

The algorithm stores the absolute properties before it begins parsing the gradable properties. This speaks to the idea that the order in which people consider properties of items

²This may also be tactile salience, auditory salience, etc., dependent on the task.

does not directly reflect the order of modifiers generated to describe those items. This assertion is supported by the evidence: In Pechmann’s famous study, he found that people began producing descriptions of an item as they scanned the contrast set, but, would produce a color descriptor *after* they had produced relative descriptors. That is, the intended referent was fixated on before the contrast set was considered, but modifiers relevant to the contrast set were uttered before the modifiers relevant to the intended referent. The order in which these two properties are analyzed does not effect the current algorithm, but if the algorithm is altered to generate as it determines content, this separation must be preserved.

The algorithm also incorporates Thomas Pechmann’s finding that people include color descriptors in 98% of their utterances to pick out an objects (Pechmann, 1989). It is assumed that a COLOR value is available in the absolute properties listed for each intended referent.

Inclusion of SIZE values is also integrated into the algorithm, as SIZE has been shown to be a common property for distinguishing reference in visual tasks. For example, of the modifiers provided for the intended referents of the TUNA corpus (Gatt, 2008), I found that 25.56% (token proportion) of the top 10 modifiers are those denote size (the rest were color modifiers).

The algorithm can derive up to three prenominal modifiers for each referent. This constraint appears to best model human output in a distinguishing reference task. Again analyzing the reference to objects available in the TUNA corpus, I found that no expressions contain more than three prenominal modifiers.³ The three modifiers selected in this algorithm are a combination of the modifiers derived from gradable properties and the modifiers derived from absolute properties.

Other kinds of properties, such as spatial relations and those that give rise to embedded prepositional phrases and relative clauses, may also be used to identify an item. The algorithm would grow in robustness by extending reference to occur in these ways. The intention of the approach outlined here is to capture the most common mechanics of reference.

³This isn’t too surprising, as the TUNA corpus only utilizes a handful of properties. But this serves as a starting point. The recall of the approach is tested in Chapter 5, and the algorithm is shown to be effective using this assumption.

3.5 Inputs and Outputs

Examples of possible inputs and outputs for the algorithm are given in Table 3.1. Each input corresponds to one or more outputs, depending on the system constraints incorporated into the microplanning component of the algorithm. The table below lists a few of the expressions that may be generated from each input.

Table 3.1: Example Input-Output Pairs

	Referent Vector	Contrast Vectors
Input:	(0,0,0) lantern color:silver luminosity:shiny	(common-value) lantern (1,0,-1) lantern taller_than wider_than
Output:	silver lantern larger silver lantern large silver shiny lantern	
Input:	(0,0,0) ball color:red material:rubber shape:spiky	(common-value) ball shape:round (1,0,0) surfboard wider_than longer_than
Output:	red ball red rubber ball spiky red ball spiky red rubber ball	

3.5.1 Inputs

I assume two inputs to my algorithm. The first is the **referent vector**. I define a referent vector \mathbf{r} to be a triple $\langle x, \mathbf{n}, \mathbf{a} \rangle$ such that \mathbf{x} is a unique and arbitrary identifier of the vector, \mathbf{n} is the entity expressed by the head noun, and \mathbf{a} is a list of attribute-value pairs expressing semantic properties for \mathbf{n} . Figure 3.3 provides an example of what this looks like.

The second input is the **contrast vectors**. I define a contrast vector \mathbf{c} to be a triple $\langle x, \mathbf{n}, \mathbf{g} \rangle$ such that \mathbf{x} is a unique and arbitrary identifier of the vector, \mathbf{n} is the entity expressed by the head noun, and \mathbf{g} is a list of gradable properties expressing relations between the intended referent and the the contrast item. The first contrast vector is required to list any gradable differences between the intended referent and the common forms for that referent. This is the **common-value** form of the intended referent. Figure 3.4 provides an example of what these look like.

The common-value form of the intended referent allows for comparison of the intended referent with what is generally true for that item. For example, if the intended referent is a small cup, but it is not smaller than anything in the context set, the common-value gradable properties in the first contrast vector are used to derive the adjective *small*. In this way, the contrast vectors provide for comparison of the intended referent to the immediate contrast set as well as to general common values of the referent.

3.5.2 Outputs

The outputs from the first two main functions of the algorithm are structures separating the two kinds of modifiers: those created from the absolute properties, the **absolute modifiers**, and those created from the gradable properties, the **gradable modifiers**. The output of the text planning function provides a structure from which many expressions may be generated; the output of the microplanning function provides a structure that corresponds to a single referring expression. This structure is then passed to the final surface realization component, which orders the absolute modifiers before generation.

3.6 System Knowledge

To analyze these absolute properties, the system must provide background knowledge that associates the head noun to common values for the given properties (if common properties exist). With this, modifiers that convey absolute properties are chosen if they are not already denoted by the head noun. For example, the modifier *round* will not be used if the head noun is *ball*. Similarly, the color constraint can be relaxed if they are definitive of the head noun; an orange will generally not be referred to as *an orange orange*.⁴

To analyze the gradable properties, the system must provide background knowledge that defines whether each item is volume-type or object-type. This approach allows the algorithm to select which contrast items are used to derive modifiers. As gradable properties are mapped to modifiers, the system is also used to determine whether each modifier is

⁴Although cases where this does happen certainly occur, they are likely examples of describing reference, not distinguishing reference.

contradictory to those already chosen.

I therefore require that the system be able to provide the following background knowledge:

1. The properties that are generally true of the intended referent
2. Whether each noun is a volume-type or surface-type entity
3. Mappings between gradable properties and modifiers
4. Which modifiers are contradictory, or antonyms

These points are discussed in greater detail in the following sections. Knowledge of common properties is used such that only those properties that are not generally true will be chosen. These properties can be provided in a dictionary, where for each entity common attribute-value pairs are listed. The process of mapping between gradable properties and their modifiers will be discussed, and a representation of the mapping for gradable size properties and modifiers is given in Appendix C. Knowledge of contradictions is used as part of the incremental process, where if a newly derived descriptor contradicts one already chosen, that new descriptor is not included in the final output.

3.7 The Referent Vector and the Contrast Vectors

In this section I discuss the two main inputs to the algorithm in detail. These are the referent vector and the contrast vectors. Example inputs for distinguishing reference of the green box (the box on the right) in Figure 3.2 are given in Figure 3.3 and Figure 3.4. The referent vector is relatively simple and is only discussed briefly. The contrast vectors provide information that is used to derive modifiers by the algorithm, and so require a detailed discussion. This is presented here.

The format of the referent vector and contrast vectors convey all the information about each item, with one item defined per line. For both kinds of vectors, the first item is an identifier, or instance name. In the present system, these identifiers have internal structure



Figure 3.2: Example Input Domain

(0,0,0) box shape:open color:green material:velvet

Figure 3.3: Example Intended Referent Vector

(common-value) box
 (-1,1,0) box taller_than wider_than longer_than
 (-1,0,0) box

Figure 3.4: Example Contrast Set Vectors

representing the locations in physical space of the referents, but this information is not currently exploited. The second item identifies the head noun to refer to the entity. Following this, the next items in each vector convey the properties that will be considered. These are absolute properties for the referent vector, and gradable properties for the contrast vectors.

The values in the attribute-value pairs of the referent vector are essentially identical to the absolute modifiers they produce. The properties of the contrast set, however, are stated as relative terms, and so must be mapped by the algorithm to specific gradable modifiers. This is done by incrementally analyzing each contrast vector in the order provided. If the intended referent and the contrast item are the same kind of entity (volume-type or surface-type, further discussed below), then the properties defined in the vector for that contrast item are used to derive modifiers.

Those modifiers that directly contradict ones that have already been selected may be used to form a new modifier that mediates between the two. For example, *medium-sized* may be generated if the modifiers *big* and *small* have been selected.⁵ Modifiers that directly contradict one another form a **strict contradiction**, where the meaning of one is opposite to the meaning of another.

Those modifiers that indirectly contradict modifiers that have already been selected are thrown out. This ensures that the first elements in the contrast set have a greater effect on the modifiers generated than the later elements. Modifiers that indirectly contradict one another form a **loose contradiction**, where the meaning of one mitigates the use of the other. *Small* and *short* form a loose contradiction. The following example illustrates the reason for this approach.

3.7.1 The Contrast Vector

Consider a scene with four animals: A pig, a duck, a mouse, and a horse. This is presented in Figure 3.5. The task is to pick out the duck. Generally speaking, mice are smaller than ducks, ducks are smaller than pigs, and horses are bigger than all of them. Now suppose

⁵This method should be improved on with further work on when people treat two (or more) entities as one contrast item. Ideally for an incremental approach, each modifier is derived from one contrast item.

that the pig and duck are standing close to one another, while the mouse and the horse are in the corner. Now suppose that the pig is smaller than the duck, which is slightly smaller than the mouse, and the horse is the smallest of all. In essence, their sizes are reversed. This means that the relation between duck and pig yields the adjective *big*, the relation between duck and horse yields the adjective *big*, and the relationship between duck and mouse yields the adjective *small*. If the contrast set vector file is ordered such that the pig is the first item from the contrast set, the mouse is the second element, and the horse is the third element, then the adjective *big* will be chosen first. When the vector corresponding to the mouse is analyzed, *small* will be chosen as a possible adjective. However, this is a strict contradiction to the already chosen *big*, and so the adjective to describe the duck becomes *medium-sized*. When the vector corresponding to the horse is analyzed, *big* again will be chosen as a possible adjective. However, this is a loose contradiction to the already chosen *medium-sized*, and as such is thrown out. In this way, the order in which the contrast set is analyzed determines the modifiers that are chosen for generation. The final description from this process will thus be something like *the medium-sized duck*.

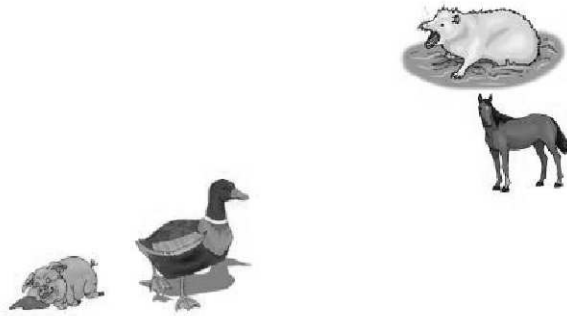


Figure 3.5: Example Scene: a Pig, a Duck, a Mouse, and a Horse

This example also serves to clarify an important point. The contrast set (provided as contrast vectors) is ordered in terms of proximity, but this ordering is here in lieu of salience.

The order in which the contrast items are considered should be in the order determined by the context. For example, in a visual task, the contrast items should be ordered according to the order people generally scan a scene. Studies have shown that it is difficult to generalize patterns of visual fixation (and therefore which items are visually salient), as different pictures yield vastly different results (Antes, 1974; Loftus and Mackworth, 1978; Rayner and Pollatsek, 1992). Until these details are better understood, proximity seems a reasonable first approach.

When the elements of salience are better understood, perhaps it will be the case that the largest items in a scene are the most visually salient, and so should appear at the beginning of the contrast set vector file. I leave this issue open-ended for now, and put in place the processes for analyzing the kind of information necessary to generate natural reference, however salience may be defined.

The gradable relations I focus on in particular are those for SIZE. Three major axes for relative size are assumed, as described in Landau and Jackendoff (1993) (see Chapter 2). This gives information about the height, width, and depth of the intended referent relative to each contrast item. Relative location, weight, and other such gradable properties may be used in distinguishing reference as well.

I also follow Landau and Jackendoff in distinguishing between surface-type entities and volume-type entities. As discussed in Chapter 2, **surface-type** entities are those that principally extend in two dimensions (such as a record), while **volume-type** entities are those that extend in all three (such as a box). These distinctions help guide the selection of modifiers to refer to an entity: Surface-type entities need only extend in one dimension to be called *big*, while volume-type entities must extend in two.

The properties captured in the contrast vectors are those that show a significantly larger or smaller measurement on an axis. Determining exactly what is significant is beyond the scope of the current work. However, Hermann and Deutsch (1976, as reported in van Deemter, 2004) have provided some insight into the nature of this. They show that differences on one axis that are much greater than differences on another are most likely to be chosen as a property in reference.

It seems reasonable to propose that gradable properties denoting SIZE be provided in the case where the difference on one axis between a contrast item and an intended referent is much larger than the difference on another axis, or much larger than a common value for the referent. Further investigation into gradable properties in particular is necessary for the creation of the contrast vectors. For now, they serve as a starting point for the algorithm.

Different gradable properties result in different modifiers. For example, when the generating axis of the intended referent is significantly larger than that of the contrast item, the relation *taller_than* is provided. This can be mapped to the adjective *tall*. As mentioned above, the selected modifiers depend on the type (volume-type or surface-type) of the referent. Therefore, in order to derive an adjective like *large* for a volume-type object, at least two dimensions must have a gradable property that expresses a larger size. For a surface-type object, extension in only one dimension is necessary. The mappings between gradable properties and modifiers are provided in Appendix C.

Again, I include as the first member of the contrast set a common-value form of the intended referent. This vector lists the gradable differences between the intended referent and a common-value form of the referent, providing a way to incorporate real-world knowledge about referents. This has the effect of ensuring, for example, *big lamp* can be generated if the lamp is particularly large, regardless of the other items in the set.

Only contrast items that are of the same type are used to derive modifiers from the contrast set. In this way, only volume-type contrast items are used to derive SIZE modifiers for volume-type intended referents (and similarly for surface-type objects). When the algorithm encounters a contrast item that is not of the same type as the intended referent, it skips to the next contrast item. When the algorithm encounters a contrast item (other than the common-value item) that is not only the same type but also has the same head noun, comparative forms can be generated (e.g., *the bigger tree*).

3.8 The Algorithm

The text planning and microplanning sections of the algorithm are given in Figures 3.6 and

CreateRefExp(A, C)

```

1: abs ← {}
2: rels ← {}
3: type ← GetType(head noun)
4: for each <att, val> ∈ referent vector A do
5:   if not CommonValue(<att, val>, head noun) then
6:     abs ← abs ∪ {<att, val>}
7:   end if
8: end for
9: for each contrast vector Ci ∈ contrast vectors C do
10:  typeCi ← GetType(head nounCi)
11:  rels ← rels ∪ FindRelProps(Ci, rels, type, head noun, typeCi, head nounCi)
12: end for
13: L ← {rels, abs, head noun}
14: return L

```

FindRelProps(C_i, rels, head_noun, head_noun_{C_i})

```

1: modifiers ← GetDescriptors(Ci, type, head noun, typeCi, head nounCi)
2: return RemoveContradictions(modifiers, rels)

```

RemoveContradictions(modifiers, rels)

```

1: new modifiers ← {}
2: for each modifier Xi ∈ rels do
3:   for each modifier Yj ∈ modifiers do
4:     if modifier Xi and Yj are a contradiction then
5:       if they form a strict contradiction then
6:         new modifiers ← new modifiers ∪ {medium-sized}
7:       end if
8:     else
9:       new modifiers ← new modifiers ∪ Yj
10:    end if
11:  end for
12: end for
13: return new modifiers

```

Figure 3.6: Text Planning: Algorithm for Distinguishing Reference

ChooseRefExp(L)

```

1: rels vals  $\subseteq$  gradable values
2: abs vals  $\subseteq$  values in absolute values
3: such that:
4:  $|\text{rels vals}| + |\text{abs vals}| \leq 3$ 
5: if  $\langle \text{COLOR}, \text{value}_{\text{color}} \rangle \in \text{absolute values}$  then
6:    $\langle \text{value}_{\text{color}} \rangle \in \text{abs vals}$ 
7: end if
8: if gradable values  $\neq \emptyset$  then
9:   if  $|\text{abs vals}| > 1$  then
10:     $|\text{rels vals}| \geq 1$ 
11:   else if  $|\text{abs vals}| + |\text{rels vals}| = 1$  and  $\langle \text{value}_{\text{color}} \rangle \notin \text{abs vals}$  then
12:     $|\text{rels vals}| = 1$ 
13:   end if
14: end if
15: for any size value  $\in$  rels vals in comparative form do
16:   choose one:
17:   size value  $\leftarrow$  base form
18:   size value  $\leftarrow$  size value
19: end for
20: re  $\leftarrow$  {rels vals, abs vals, head noun}
21: return re

```

Figure 3.7: Microplanning: Algorithm for Distinguishing Reference

3.7. The first main function is given as `CreateRefExp`. This takes as input the intended referent vector (*A*) and the contrast vectors (*C*). Lines 1 and 2 set the variables to store the absolute properties (*abs*) and gradable modifiers (*rels*). Line 3 sets a variable (*type*) to the type value for the head noun of the intended referent (for example, *volume-type*). In lines 4 through 8, the function iterates through the absolute properties of the intended referent vector and stores those properties that are not a common-value form in a list. In lines 9 through 12, the function iterates through the contrast vectors and calls to `FindRelProps` to return the modifiers that can be derived from these vectors. The final structure (*L*) is a list with the ordered gradable modifiers, the absolute properties, and the head noun of the intended referent.

This structure is passed to the second main function, `ChooseRefExp`. Lines 1 through 10 select the specific gradable modifiers (*rels vals*) and absolute modifiers (*abs vals*) to be used in the final referring expression. Line 4 imposes the constraint that no more than three modifiers are selected, lines 5 through 7 impose the constraint that a color modifier is included if it exists, and lines 8 through 14 impose the constraint that a size modifier is included if it exists and there is no color modifier or more than just a color modifier is desired. Lines 15 through 19 impose the constraint that a size value may appear in either comparative form or base form. This structure (*re*) contains the ordered gradable modifiers, the unordered absolute modifiers, and the head noun that may be passed to the surface realization stage.

The first step in the algorithm is therefore to parse the input vector and check for common values, storing those values that are not already defined for the head noun (e.g., if `SHAPE:'round'` is provided as an input property, *round* will not be used if the head noun is *ball*). The referent vector thus provides the basis for generating a description that something is *red*, or *wooden*.

The second step in the algorithm is to parse the gradable properties represented in the contrast vectors, which convey `SIZE` relations. These are parsed in the order they are provided. The contrast vectors thus provide the basis for generating a description that something is *tall*, *thin*, or *big*. With this approach, the relative size of the referent to each

surrounding member is used instead of exact measurements on each item.

As the analysis of the contrast set begins, so too does the construction of all the possible referring expressions. With the absolute properties of the item already stored, the ordering of gradable properties begin to define the final output. The algorithm iterates through each item in the contrast set. For each contrast item, if its type is the same as the intended referent type, a modifier can be derived from its listed gradable properties. The algorithm then proceeds to the next contrast item.

Contradictory information is handled by the `RemoveContradictions` function, which takes as input the modifiers defined by the current contrast item, and the list of modifiers that are already part of the referring expression structure being created. It returns a new list of modifiers that is stripped of any contradictory information. If any of the newly considered modifiers form strict contradictions to those in place (for example, if *small* has already been chosen, and *large* is encountered), the modifier *medium-sized* replaces the contradictory modifier. If any of the newly considered modifiers form loose contradictions (for example, if *small* has already been chosen, and *short* is encountered), they are simply not included in the new list. The selected modifiers are then appended to the referring expression structure.

After the contrast set has provided for the generation of these modifiers, the absolute property tuples and gradable modifiers are stored. A structure with all of these modifiers along with the head noun are then passed in a structure to the second major component of this algorithm. This appears in Figure 3.7 as `ChooseRefExp`.

`ChooseRefExp` selects which modifiers are included such that the total number of modifiers selected is ≤ 3 . Evidence shows that there is a clear preference in natural language for including COLOR and SIZE modifiers, and so preference is given for inclusion of these properties. If a value for COLOR is passed from `CreateRefExp`, this value is selected. To create a referring expression with more than one modifier, if a SIZE value is passed from `CreateRefExp`, then it is selected. To create a referring expression with three modifiers, the longest expression this algorithm generates, a third modifier may be selected from either the gradable modifiers or the absolute modifiers.

The final referring expression structure can then be passed to a Surface Realizer, where

the modifiers conveying absolute properties can be ordered using a type system. The referring expression structure enforces a separation between absolute modifiers and gradable modifiers. This way, the Surface Realizer can easily separate out and order only the absolute modifiers.

The algorithm calls to three smaller functions to be provided by the system, `CommonValue`, `GetType`, and `GetDescriptors`. `CommonValue` checks if the attribute-value pair is not already denoted by the head noun. If the value is denoted, the function returns `TRUE`. If it is not denoted, the function returns `FALSE`. `GetType` returns whether the referent is volume-type or surface-type. `GetDescriptors` takes as input the list of relations and the item’s type value, and maps those relations to modifiers as defined in the modifier-relation map and outlined in Appendix C. If an intended referent and the contrast item to which it is being compared are of the same head noun, a comparative form is generated. If this contrast item is the common-value form of the intended referent, a comparative form is not generated.

3.9 Algorithm Summary

The sections of this algorithm introduced here can be schematized as follows:

System Knowledge: Common value representations of referents and their types (volume-type or surface-type), mappings between gradable properties and modifiers, knowledge of antonyms

Inputs: A referent vector and contrast vectors

Computation: Non-incremental analysis of absolute properties to derive absolute modifiers; incremental analysis of contrast items and their corresponding gradable properties to derive gradable modifiers

Outputs: A referring expression structure to be realized by a Surface Realization module using a modifier type system

3.10 Example

I will work through one specific example in order to illustrate how the algorithm works. Consider the data given above, with the referent vector in Figure 3.3 and the contrast vectors in Figure 3.4. The algorithm takes as input the referent vector and the contrast vectors. Some sample input and output are given in Table 3.1. The modifier mapping and contradictions file should be provided by the system.

The first step in the algorithm is to acquire the head noun and determine its type. This is the second element in the referent vector file. In this example, the head noun is *box* which is a volume-type object.

The next step is to gather the attribute-value pairs defined in the referent vector. These follow the head noun. For each attribute-value pair, the algorithm checks if the value of that attribute is generally true of the head noun. If it is not, the attribute-value pair is stored. In this example, none of the attribute-value pairs are generally true of the head noun. And so, SHAPE:‘open’, COLOR:‘green’, and MATERIAL:‘velvet’ are stored in a list.

Once these absolute values are stored, the algorithm moves on to incremental analysis of the contrast set. For each contrast item that is of the same type (volume-type), the algorithm gathers the gradable properties. In this example, the first contrast item, the common-value comparison for the intended referent, contains no information: There are no common values for a box’s size. The algorithm then moves on to the next contrast vector. This provides the relations *taller_than*, *wider_than*, and *longer_than*.

The algorithm then maps these gradable properties to a modifier (or modifiers), utilizing the type information to guide how many relations are necessary to create a more general modifier, like *big*. If the two entities being compared have the same head noun, and the contrast item is not the common-value vector, then comparative forms of the modifiers are created – for example, *shorter* and *bigger*. Superlative forms can be created when a contrast item is a set. This problem is left unaddressed for now.

In this example, the intended referent is defined as taller than, wider than, and longer than the first contrast item. Where the corresponding adjectives *tall*, *wide*, and *long* could

be returned, the modifier mapping provides for the adjective *big*. This is because all three relations indicate a larger measurement. Had the vector provided the information *taller_than*, *thinner_than*, *longer_than*, the adjectives *tall*, *thin*, and *long* would be returned. In this way, gradable properties for a single contrast item may be mapped to one, two, or three modifiers.

The next step is to see if the modifier(s) found contradict any of the modifiers chosen previously. For example, if the algorithm has already chosen the adjective *small*, it should not now choose the adjective *big*. As this is the first modifier in the example, there are no contradictions and *big* is added. This can be realized as *big* or *large*, and because the contrast item that generates this modifier has the same head noun, *bigger* or *larger*.

The algorithm then moves on to the next contrast item. It has the same type value, so the relations can be analyzed. But, there are no relations: The contrast item's size does not significantly differ from the intended referent in any way. No gradable properties are analyzed, and no new modifiers are gathered.

At this point, the text planning part of the algorithm is done. The referring expression structure returned is a list composed of the two smaller lists of gradable modifiers and absolute modifiers followed by the head noun and type. For this example, the structure is given in Figure 3.8.

The second part of the algorithm then selects which modifiers it will use. Because there is no common COLOR value for a box, inclusion of a COLOR modifier is mandated. If more than one modifier is desired, the inclusion of a SIZE modifier is mandated. Possible structures for this is given in Figure 3.9.

The structure returned in `CreateRefExp` provides the means for realizing a variety of referring expressions, once the absolute properties are ordered by the surface realization component. This is discussed in Chapter 4. The possible expressions that can be derived for this example are given in Figure 3.10.

```
[[['big'], [COLOR:'green', SHAPE:'open', MATERIAL:'velvet'], 'box']]
```

Figure 3.8: Text Planning Example Referring Expression Structure

```

      [], ['green', 'box']           [['big'], ['green'], 'box']
    [['large'], ['green'], 'box']   [['big'], ['green', 'velvet'], 'box']
  [['large'], ['green', 'velvet'], 'box']  [['large'], ['green', 'open'], 'box']
    [['big'], ['green', 'open'], 'box']

```

Figure 3.9: Microplanning Example Referring Expression Structures

```

      green box           big green box
    large green box     big green velvet box
  large green velvet box  large green open box
    big green open box

```

Figure 3.10: Example Generated Referring Expressions

3.11 Example Implementation

item:	i-048271
class:	box
shape:	open
color:	green
material:	velvet
height:	4 in.
length:	4 in.
width:	4 in.

Figure 3.11: Example ILEX Input

To outline how this algorithm may be implemented in an existing system, I will discuss the NLG system ILEX (O'Donnell, 2000). This system generates information on museum objects, and so is a suitable example of the kinds of reference my own algorithm generates.

```

(say example-1
  :is propose
  :proposition leaving
  :relevant-roles ( (leaving Actor)
                    (causation Head
                      Dependent)
                    (arrival Actor))
  :identifiable-entities (John Mary))

```

Figure 3.12: Speech Act Input for WAG (O'Donnell, 2006: 7)

The implementation I outline serves as a general example. Much more work is necessary in order to integrate the approach I outline in this thesis into any system.

ILEX was developed using an exhibit database from the National Museum of Scotland. The information that serves as input to this system is represented in database format, as shown in Figure 3.11. ILEX uses a mix of canned text and the sentence generation system WAG (O'Donnell, 1996) in order to generate descriptions of items from these database structures.

The input specification for the WAG sentence generator is a speech act. An example of this is shown in Figure 3.12. The speech act structure can then realized as a sentence; in this example, “Mary left because John arrived”.

To implement the proposed algorithm into such a system, the Text Planning component of ILEX first must construct a structure from the database specification (see Figure 3.11) to feed as input to the algorithm. This would correspond to the referent vector in Figure 3.3. Similar database entries could be used to construct the contrast vectors in Figure 3.4.

The content determination function of the algorithm I have outlined then may plan the text for reference to this object, as discussed above. The microplanning section of the algorithm maps the gradable modifiers, selected absolute properties, and head noun to a linguistic structure specifying the head noun and modifiers. This structure may be mapped directly to the identifiable-entities structure in the speech act formalism used by ILEX. An example is given in Figure 3.13.

ILEX then passes this structure through the WAG Surface Realizer, where the absolute properties can be ordered using the modifier type system. From this, a final referring expression is generated. In this example, the phrase *large green open box* may be created.

```
(say :identifiable-entities ( (gradable-properties large)
                              (absolute-properties green open)
                              box))
```

Figure 3.13: Identifiable Entity for Surface Realization

3.12 Discussion

The approach to naturalness taken here differs in several specific ways from the approaches taken to-date. These are:

1. Incremental examination of the contrast set (not incremental examination of attribute-value pairs)
2. Descriptors split into those inherent to the intended referent (absolute properties) and those derived from the contrast set (gradable properties)
3. Ordering of descriptors based on the order of the contrast set and a surface realization component (not by a predefined list of properties)
4. Mandated inclusion of a color modifier if the color modifier is not definitive of the object
5. Inclusion of a size modifier if it exists and more than a color modifier is desired

The first item is integral to the generation of natural reference. The second seems well-supported by what we know so far about human generation of referring expressions, and falls out from the first. The third is a reasonable approach for handling the prior two items. The last reflects the findings from corpus and psychological studies.

3.13 Summary

The approach to generation outlined here is a departure from much of the work done in the field, most notably the Incremental Algorithm (Dale and Reiter, 1995) and the extensions to this algorithm that assume the same fundamental input (Horacek, 1997; Stone, 2000; van Deemter, 2002; Gardent, 2004). As stated above, the notion of utilizing as input a predefined ordering of properties, where the ordering of the input determines the ordering of the output, is abandoned. In its place, two vectors with different kinds of properties serve as input. The ordering of the modifiers derived from the first vector is dependent on a surface realization component. The ordering of the modifiers in the second vector is dependent on the ordering of the actual contrast items.

Mirroring the general flow of natural language generation, the algorithm outlined here provides for integration into generation systems. The functions introduced here instantiate the incremental approach reported by Pechmann (1989), using the two basic kinds of modifiers noted and discussed by many researchers (Whorf, 1945; Pechmann, 1989; Dale and Reiter, 1995; Krahmer et al., 2003; van Deemter, 2004). The model developed from these ideas is shown to be capable of generating a wide range of natural sounding referring expressions in Chapter 5.

Chapter 4

ORDERING MODIFIERS FOR REFERENTS**4.1 Introduction**

In this chapter, I establish and evaluate a type system that can be used to order modifiers in a Surface Realizer. The application of this type system is not limited to the generation of distinguishing reference, but also to describing reference and any noun phrase with modifiers prenominally. Ordering modifiers based on the types introduced here completes the algorithm introduced in Chapter 3. Predictions of prenominal modifier ordering based on these classes are shown to be robust and accurate.

The algorithm that I have developed for distinguishing reference may order the absolute properties according to the type ordering outlined in Table 4.6. Any other noun phrase with prenominal modifiers may utilize this type system as well. Ordering modifiers based on these types is a relatively simple task. This chapter therefore takes a break from the discussion of the algorithm, and outlines how the type system is created.

The work here diverges from the approaches commonly employed in modifier (usually adjective) classification by assuming no underlying relationship between semantics and syntax or morphology and syntax. Although these certainly may exist, the basis for determining modifier classes here does not rely on any such proposed relationship. This provides a starting point for further research by outlining some main ideas on how to approach the problem.

The chapter begins with a discussion of the relationship between modifier ordering and referring expression generation. I then turn to a presentation of the ideas behind the modifier type system developed here. Following this, I present the methodology used in the current study, discussing the corpus involved and the basic modules used in the process. This is followed with a discussion of preliminary results, and an outline of possible future work in

the area.

4.2 *The Model and the Problem*

Generating referring expressions in part requires generating noun modifiers. These modifiers must appear in an order that sounds natural and mimics human natural language use. For example, consider the alternation below as given in Vendler (1968).

- (4) big beautiful white wooden house
- (5) ?white wooden beautiful big house

- (6) comfortable red chair
- (7) ?red comfortable chair

- (8) big rectangular green Chinese silk carpet
- (9) ?Chinese big silk green rectangular carpet

Figure 4.1: Grammatical Adjective Alternations (Vendler, 1968: 122)

Some combinations of modifiers before a noun sound perfectly natural, while other combinations – using the very same modifiers – sound marked, or at least less common.

Almost all referring expression generation algorithms rely on the availability of a pre-defined ordering of properties (Dale and Reiter, 1995; van Deemter, 2002; Krahmer et al., 2003). This requires that for every referent, an ordered listing of all the modifiers that can apply to it must be created before referring expression generation begins. This provides a direct mapping between the ordering of attribute-value pairs in input and the prenominal modifier ordering in output.

Deriving the ordering of attribute-value pairs is not a simple task, but rather one that requires intricate work for every referent, for every combination of modifiers. Ordering individual modifiers by hand can lead to accurate generation results. However, proceeding with generation in this way seems to miss two clear generalizations:

1. Not all modifiers have equally stringent ordering preferences.

2. Modifiers can be grouped into types indicative of their ordering preferences.

The difficulty with capturing the ordering of modifiers stems from the problem of data sparsity. In the example above, the modifier *silk* may be rare enough in any corpus that finding it in combination with another modifier, in order to create a generalization about its ordering constraints, is nearly impossible. Malouf (2000) examined the first million sentences of the British National Corpus and found only one sequence of adjectives for every twenty sentences. With sequences of adjectives occurring so rarely, the chances of finding information on any particular modifier is small. The data is just too sparse.

Because of this, the process of classifying modifiers for predicting order is especially problematic. Most approaches assume an underlying relationship between semantics and prenominal position (cf. Whorf, 1945; Ziff, 1960; Martin, 1969a and 1969b; Bever, 1970; Danks and Glucksberg, 1971; see Chapter 2 for more). These approaches can be characterized as predicting modifier order based on degrees of semantic closeness to the noun. This follows what is known as **Behaghel's First Law** (Behaghel, 1930):

Word groups: What belongs together mentally is placed close together syntactically.

(Clark and Clark, 1977: 545)

However, there is disagreement on the exact qualities that affect position. These theories are also difficult to implement, as they require determining the semantic properties of each modifier used, relative to each context in which it occurs. If a modifier classification scheme is to be implemented, it should be robust enough to handle a wide variety of modifiers and flexible enough to allow different orderings.

4.3 *Towards a Solution*

In an attempt to create a flexible system capable of predicting a wide variety of orderings, the approach to modifier classification presented here is based on modifier distribution. Modifiers are classified by where they tend to appear prenominally, in a system that lends itself to bootstrapping for unseen modifiers. This classification scheme allows rigid as well

as more loose orders (compare *big red ball* and *?red big ball* with *white floppy hat* and *floppy white hat*). It is not based on any mapping between position and semantics, morphology, or phonology, but does not exclude any such relationship in the classification: The classification scheme builds on what there is direct evidence for, independent of why each modifier appears where it does.

The first step in this approach is to use a modifier-rich corpus. For this, I use the massive Google Web 1T 5-gram corpus (Brants and Franz, 2006). The second step is to use a lexical database to leverage information. For this, I use WordNet (Miller, 2006), which provides information on whether words have an adjective sense, noun sense, verb sense, or adverb sense. This was chosen instead of a parser or a POS (Part-of-Speech) tagger because the corpus contains collections of N-Grams, not sentences. Both POS taggers and parsers rely on surrounding tag contexts and joint conditioning on multiple consecutive words in order to derive tags (Toutanova et al., 2003), but WordNet allows analysis on a word-by-word basis, which is helpful when training on N-Grams. WordNet also provides information on any word that may be used as a prenominal modifier by marking it as having an adjective sense. This is regardless of whether the word would generally be tagged as a verb or noun, and so facilitates the extraction of words that can be used as modifiers. So, for example, *square metal plate* can be extracted as *Adj Adj Noun*, where the prenominal positions of *square* and *metal* may be used to determine position preferences for those modifiers, but *plate metal* will only be extracted as *Noun Adj* or *Noun Noun*, and *plate* will not be stored as a modifier.

The decision not to use the British National Corpus or the Wall Street Journal, which are common in most NLP tasks, was due to their relatively low frequency of adjectives, as discussed above. Using a corpus rich in prenominal modifiers ensures that some generalizations about the distribution of modifiers can be made. It also provides information on a greater number of modifiers, allowing more modifiers to be classified.

Some basic assumptions are made as a starting point. The approach here builds on an idea that nouns will rarely have more than four preceding modifiers. This allows me to assume four possible modifier positions and construct probabilities of modifier position

based on where different modifiers occur. The algorithm introduced in Chapter 3 only generates up to three modifiers to best model what people tend to do; this corpus study looks at up to four in order to develop a more complete model of ordering.

4.3.1 *Classifying Modifiers*

The method used to classify modifiers is as outlined here:

1. Select or construct a modifier-rich corpus.
2. Extract all noun phrases that consist of a head noun preceded by more than one modifier, using WordNet to find all consecutive modifiers immediately followed by a noun.
3. Store the counts and relative position of each modifier.
4. Convert this into probabilities in vector file format.
5. Classify modifiers based on the positions that have the highest probabilities.

Examples of the files in Step 3 and 4 are given in Table 4.1 and Table 4.2.

Table 4.1: Example Modifier Classification Intermediate File: Step 3

popliteal	one	2023				
resonant	two	87	three	82	four	16
unscientific	one	521	two	59	three	59
omnipotent	three	84				

Table 4.2: Example Modifier Classification Intermediate File: Step 4

popliteal	one	1.0				
resonant	two	0.47	three	0.44	four	0.09
unscientific	one	0.82	two	0.09	three	0.09
omnipotent	three	1.0				

4.4 Materials

To create the training and test data, a corpus and a lexical database were used. Several programs were constructed to train and analyze the information provided by these data. Below, I outline the details of these data and code modules.

4.4.1 Data

Google’s Web 1T 5-gram – (Brants and Franz, 2006). A collection of N-Grams (1-Grams, 2-Grams, 3-Grams, 4-Grams, and 5-Grams) from web pages made available by the Linguistic Data Consortium. To create this corpus, Google processed 1,024,908,267,229 words of running text from websites. Frequency counts were calculated for N-Grams that appear at least 40 times. Words that appear less than 200 times were discarded, yielding 13,588,391 unique word types. These form a total of 1,176,470,663 5-Grams.

WordNet – (Miller, 2006). This is a large lexical database of English, where words are grouped into sets of “cognitive synonyms” or **synsets**. This can be used to mark words as adjectives and nouns.

4.4.2 Code Modules

The following five components were developed (in Python) for this project.

Modifier Extractor – This program takes as input the Google 5-gram corpus, and returns all the noun phrases immediately preceded by prenominal modifiers, along with their frequency counts. These frequency counts are used for training, but not for testing. This program only returns those N-Grams that are composed of consecutive modifiers immediately followed by a noun.¹ Nouns that are preceded by only one modifier are not considered, as these phrases do not provide insight into the ordering of modifiers prenominally.

¹That is, this program returns 3-Grams, 4-Grams, and 5-Grams extracted from the 5-Grams.

input: Google 5-Gram Corpus

output: List of modifier-rich noun phrases and their frequencies

Modifier Organizer – This program takes as input lists of noun phrases with prenominal modifiers, and filters some of the words commonly tagged as modifiers by WordNet. These are individual letters, words that have a greater amount of senses as a noun or verb than as an adjective, and words that have adverbial and prepositional readings. A list of what was filtered is available in Appendix B. This returns a vector with frequency counts for all positions in which each observed modifier occurs.

input: Modifier-rich noun phrases and their frequencies

output: Vector file with distributional information for each modifier position

Modifier Classifier – This program takes as input a vector file with distributional information for each modifier’s position, and from this determines the classification for each modifier.

input: Vector file with distributional information for each modifier position

output: Vector file with each modifier associated to a class

Prenominal Modifier Ordering Predictor – This program takes as input two files: A vector file with each modifier associated to a class, and a list of modifier-rich noun phrases (for testing). The vector file assigns a class to each modifier seen in the testing data, and predicts the ordering for all the modifiers that appear prenominally. A discussion of the ordering is given below. This program then compares its predicted ordering of modifiers prenominally to the observed ordering of modifiers prenominally. It returns Precision and Recall values for its predictions.

input: Vector file with each modifier associated to a class, list of modifier-rich noun phrases

output: Precision and Recall for modifier ordering predictions

4.5 Method

4.5.1 Classification Scheme

As discussed above, I assume four primary modifier positions. The longest noun phrases for this experiment are thus those with five words: Four modifiers followed by a noun. This assumption fits well with Google’s corpus, whose longest N-Grams are 5-Grams.

four	three	two	one	
small	smiling	white	fuzzy	bunny

Figure 4.2: Example 5-Gram with Prenominal Positions

The process for determining each modifier’s type is as follows:

- The frequency of each modifier in each position is counted.
- This is turned into a probability over all positions.
- All position probabilities ≤ 0.25 (baseline) are discarded.
- Those positions that remain determine the modifier type.

To calculate modifier position for each phrase, counts were incremented for all feasible positions. For example, in the phrase *clean wooden spoon*, the adjective *clean* was counted as occurring in positions two, three, and four, while the adjective *wooden* was counted as occurring in positions one, two, and three. The classification that emerges after applying this technique to a large body of data gives rise to the positional preferences of each modifier. In this way, a modifier with a strict positional preference can emerge as occurring in just one position; a modifier with a less strict preference can emerge as occurring in three.

Using this system, there are nine derivable modifier types, listed in Table 4.3. The frequencies of all extracted groupings of prenominal modifiers are shown in Table 4.4. The frequencies of the extracted types are shown in Table 4.5.

Table 4.3: Modifier Types

Type 1: one	Type 6: two-three
Type 2: two	Type 7: three-four
Type 3: three	Type 8: one-two-three
Type 4: four	Type 9: two-three-four
Type 5: one-two	

This table shows that the amount of phrases with four prenominal modifiers is an order of magnitude higher than phrases with three prenominal modifiers, which is unexpected. On examination of the data, it becomes clear that the reason for this is these 5-Grams include lists of nouns, which can be tagged as four modifying nouns followed by a head noun. For example, *Chinese Polish Portuguese Romanian Russian* was observed 115 times. This may be used as a noun preceded by modifiers, but most likely it is a list. That is, the 5-Grams include many sequences of words in which the final one has a use as a noun and the earlier ones have uses as adjectives. This speaks to a weakness in using an N-Gram corpus for this kind of data: N-Grams are isolated from their surrounding context. The rightmost noun in each N-Gram is not guaranteed to be a head noun, and neither a parser nor a POS tagger could overcome this problem. However, the sheer amount of information provided by the corpus may minimize the effects of this noise.

Table 4.4: Frequency of Prenominal Modifier Amounts

Number of	Percentage	
Prenominal Modifiers	Count	of Data
two	117416	81.96%
three	2459	01.72%
four	23391	16.33%

Table 4.5: Modifier Type Distribution

Type	Position	Count	Percentage
1	one	253	03.76%
2	two	185	02.75%
3	three	185	02.75%
4	four	180	02.68%
5	one-two	203	03.02%
6	two-three	1660	24.67%
7	three-four	135	02.01%
8	one-two-three	2002	29.75%
9	two-three-four	1927	28.63%

Modifiers of Type 8, the class for modifiers that show a general preference to be closer to the head noun but do not have a strict positional preference, make up the largest portion of the data. This is followed in proportion by modifiers of Type 9, the class for modifiers that show a general preference to be farther from the head noun, but do not have a strict positional preference. This reflects the large proportion of phrases extracted with only two prenominal modifiers.

Examples of Type 8 are *golden* and *cyclical*, and examples of Type 9 are *ornamental* and *copious*. With these defined, *ornamental golden crown* is predicted to sound grammatical, while *?golden ornamental crown* may sound more marked. *Copious cyclical patterns* is predicted to sound grammatical, while *?cyclical copious patterns* may sound more marked.

Using this classification scheme, positional preferences are learned from the corpora. Modifiers are assigned to the classes dependent on where they appear. This system is then used to predict the ordering of modifiers prenominally, given an unordered list corresponding to modifiers that appear in a given noun phrase. The ordering assumed for each type is given below. The proposed ordering constraints for these types are listed in Table 4.6.

Table 4.6: Proposed Modifier Ordering

Type	Position	Generated Before Type							
1	one	2	3	4	5	6	7	8	9
2	two	3	4	6	7	9			
3	three	4	7						
4	four								
5	one-two	2	3	4	6	7	8	9	
6	two-three	3	4	7	9				
7	three-four	4							
8	one-two-three	4	6	7	9				
9	two-three-four	4	7						

Classification in this way illustrates the idea that some modifiers exhibit stringent ordering constraints, while others have more loose constraints. That is, some modifiers may always appear immediately before the noun, while others may generally appear close to or far from the noun.

Note that using this classification scheme, the ordering of modifiers that belong to the same type is not predicted. This seems to be reflective of natural language use: For example, both *domestic* and *organic* are predicted to be Type 6, the type for modifiers that occur in positions two and three. This seems reasonable: Whether *domestic organic beer* or *organic domestic beer* is a more natural ordering of prenominal modifiers seems at least debatable. The freedom of intra-type positioning allows for some randomization in the generation of prenominal modifiers, where other factors may be used to determine the final ordering. This seems comparable to natural language use.

To test this system, 10-fold cross-validation was used on the extracted N-Gram corpus. The held-out data was selected as random lines from the corpus, with a list storing the index of each selected line to ensure no line was selected more than once. In each trial,

modifier classification was learned from 90% of the data and the types derived were used to predict the ordering of modifiers preminally on the remaining 10%.

4.6 Results

In order to test how well the proposed system works, all the modifiers preceding each noun were stored in unordered groups. All possible orderings were predicted based on the stored classes of the modifiers. For the baseline, a random ordering was predicted, except between identical modifiers (e.g., *pink* and *pink*), which were grouped together.

In the context of classification tasks, precision and recall measurements provide useful information of system accuracy. Precision, as defined in (7), is the number of true positives divided by the number of true positives plus false positives. This is calculated here as $\text{tp}/(\text{tp} + \text{fp})$, where tp is the number orderings that were correctly predicted, and fp is the number of orderings not correctly predicted. This measure provides information about how accurate the modifier classification is. Recall, as defined in (8), is the number of true positives divided by the number of true positives plus false negatives. This is calculated here as $\text{tp}/(\text{tp} + \text{fn})$, where tp is the number of orderings that were correctly predicted, and fn is the total number of orderings that could not be predicted because a modifier was out-of-vocabulary. This measure provides information about the proportion of modifiers in the training data whose ordering could be classified.

$$(10) \text{ Precision} = \text{tp}/(\text{tp} + \text{fp})$$

tp = the number of orderings correctly predicted

fp = the number of orderings not correctly predicted

$$(11) \text{ Recall} = \text{tp}/(\text{tp} + \text{fn})$$

tp = the number of orderings correctly predicted

fn = the number of orderings that could not be predicted because a modifier was out-of-vocabulary

The values given are averages over each trial. The standard deviation for each average is given in parentheses. In the baseline, there are no false negatives, so recall cannot be

compared. That is, a random ordering is assigned for all modifiers in the test data, and therefore 100% of the test data is covered. Results are shown in Table 4.7.

Table 4.7: Precision and Recall for Prenominal Modifier Ordering

	Type Precision	Type Recall	Token Precision	Token Recall
System	78.77% (0.45)	98.38% (0.18)	79.23% (2.75)	99.27% (0.12)
Baseline	42.88% (0.32)	-	36.31% (4.26)	-

On average, 11,121 modifiers were classified in each trial, with only between 150 to 215 modifiers outside of the vocabulary for each trial. The system also produced an average of 2.3 possible orderings for each collection of prenominal modifiers. This reflects the idea that a few orderings may be possible for a given grouping of modifiers.

As can be seen, the proposed system works well. With a type precision of 78.77% and a token precision of 79.23%, most modifiers were correctly ordered prenominally. The recall scores are even higher, with almost all modifier orderings covered.

4.7 Discussion

The system precision and recall here are striking. With this system in place, perhaps unknown modifiers could be classified based on the known classification of the surrounding modifiers, in a lexical acquisition task.

Example: grey shining metallic chain
 three_four unknown one_two head_noun

Given its position and the classes of the surrounding modifiers, **unknown** could be **two_three**.

Ordering modifiers based on this classification system creates orders seen in natural language. It follows that a Surface Realization module could use this system to order

modifiers prenominally. Out-of-vocabulary modifiers may be learned over time, and the system could default to assuming they are of Type 8 (occurring in position one-two-three), the largest class.

The usage proposed here is one where generating distinguishing reference utilizes the modifier classes and modifier mappings to order modifiers that denote absolute properties. This is the final step in generation, and completes the algorithm introduced in Chapter 3. Describing reference can use this system as well, to order all modifiers that have been determined.

To complete the schematization of the algorithm for distinguishing reference begun in Chapter 3, the entire algorithm can be summarized as follows:

System Knowledge Common value representations of referents and their types (volume-type or surface-type), knowledge of antonyms, mappings between gradable properties and modifiers, a modifier type system

Inputs A referent vector and contrast vectors

Computation Non-incremental analysis of absolute properties to derive absolute modifiers; incremental analysis of contrast items and their corresponding gradable properties to derive gradable modifiers; a modifier type system to order absolute modifiers

Outputs Natural sounding referring expressions

Before turning to evaluation of the algorithm, a note on the language dependency of the prenominal modifier ordering proposed here is necessary. This approach is written specifically for English, using sources for prenominal modifier ordering from Dutch, German and English. It is not just Indo-European-centric, but West-Germanic-centric.

According to Greenberg (1963), the majority of languages with dominant VSO order place adjectives after the noun. Greenberg asserts that the order of lexical items in these phrases frequently mirrors the order in English, although this is applied specifically to the order of demonstrative-numeral-adjective, and not to orderings of different kinds of adjectives. It may be the case that these phrases mirror English order, such that gradable

adjectives still appear farther from the head noun, and as such are generated after the absolute adjectives.

The computational model developed in this thesis still works in this case, but obviously, the referring expression structure would need to be mirrored to reflect the different ordering. However, if phrases are found (in English or in other languages) where gradable modifiers appear closer to the head noun than absolute modifiers, then it may be the case that the two kinds of modifiers are not as distinct as this algorithm proposes. In this case, more work is necessary to advance generation for natural reference in other languages.

Chapter 5

EVALUATION

5.1 Introduction

This chapter presents the evaluation of the algorithm proposed in Chapter 3. Section 5.2 addresses the goals of the algorithm, and so motivates the evaluation here. Section 5.3 reviews the materials used. Section 5.4 outlines the method. Section 5.5 presents the results from the evaluation, and compares these to results from the Incremental Algorithm on the same task. I follow the approach taken in Viethen and Dale (2006), and calculate only recall for the algorithm. This is necessary given the free-form nature of the evaluation task, which precludes calculating precision. Section 5.6 discusses the significance of the results and areas for further improvement.

5.2 Purpose

The algorithm introduced in this thesis implements a strategy for producing referring expressions similar to those that humans produce. As discussed in previous chapters, the algorithm has been constructed using the evidence on how people process information and the final expressions they produce. These ideas are implemented by initially storing absolute properties, incrementally analyzing gradable properties, and only selecting particular kinds of properties for the final referring expression. The expected result from this is the production of referring expressions identical to those found in natural language, given the same input domain.

If the algorithm does produce natural distinguishing reference, then it will be able to generate most of the referring expressions found in natural language for a distinguishing reference elicitation task. The following sections show that this is in fact true.

5.3 *Materials*

Stimuli. Subjects were presented with graphical depictions of people, animals, and objects. Some were presented as drawings, and some were presented as photographs. The pictures were composed of two or more entities, where each had at least two distinct properties.

Presentation of stimuli. Four web pages were designed to display the stimuli, with five pictures presented per page, yielding a total of 20 pictures. Each picture was presented with a sentence below, where noun phrases were replaced by text boxes to guide the intended referent. For example, “_____ is on the left side and _____ is on the right side”. The blanks here represent writable text boxes. Each picture was used to elicit between 1 and 3 referring expressions.

Delivery of stimuli. Amazon’s Mechanical Turk (Amazon, 2008) program was used to run the evaluation. This program is web-based, where users can choose to take a variety of tests for minimal reward. A screenshot of this test is available in Appendix D. The test was presented on the web site with my name, the expiration date of the test, the time allotted, and the reward for each test (\$0.01). Due to the nature of the Mechanical Turk program, users are anonymous. Each test was programmed to close after one hour, and a timer in the upper left corner notified subjects of their progress. Subjects were instructed to help identify the objects in the pictures by typing into the text boxes. A “Submit” button was placed at the bottom of each page, and so results were submitted for every five pictures.

All pictures from this test are available in Appendix A.

5.4 *Method*

5.4.1 *Elicitation*

A total of 41 referring expressions were elicited from 70 participants, yielding 2,870 individual referring expressions. From this, all expressions for reference to objects were extracted,

yielding 28 referents and 1,960 speaker responses.

A pilot study with no instructions returned results where subjects did not fully complete sentences. For example, *red* was supplied in a slot where *the red lantern* would have made the sentence grammatical. Based on this, the instructions were written asking subjects to “Help identify the objects, people, and animals. Fill in the blanks as if reading a story book to a five-year old. Please be descriptive and clear.” The decision to ask people to fill in blanks as if reading to a child was to control for intended audience and to inconspicuously avoid free-form vulgarity that may arise from anonymity; this move, however, yielded many descriptions that were unexpected (*Daddy’s boots* and *the pretty lady*).

Once the results were in, typos were corrected and spelling was normalized.¹ The data was analyzed and counts for every word were collected, as well as counts for every phrase.

5.4.2 Filtering

Two referents in the test are sets, and so were not included in the evaluation (the bags in Cell 8 of Appendix A and the yellow boots in Cell 9 of Appendix A). A minority of the elicited expressions have syntax not covered by either algorithm: embedded prepositional phrases and relative clauses. These were excluded as well. Two possessive phrases were also found, and excluded. The expressions that were filtered do not bias the data towards my own algorithm; neither the proposed algorithm nor the Incremental Algorithm was developed to generate these kinds of linguistic phenomena.

Once analysis started, it became clear that three more images had not made the intended referents clear; these results were also excluded from the study. Reference to the wrong entity (e.g., to something else in the scene) were also discarded. These steps produced a final test set of 19 referents, with 1,162 individual referring expressions.

5.4.3 Calculation

The elicited human expressions displayed a wide variety of values, and a wide variety of selected properties. For example, some people described a ball as *red* while others described

¹E.g., *multi coloured* became *multi-colored*, while *multi color* became *multi-color*.

it as *orange* (see Table 5.3). This kind of variation presents a problem for calculating recall: With *red* provided as input to the algorithm, no expressions with *orange* can be predicted by the algorithm, even if the expression contains the exact same attributes.

Because both the algorithm proposed in this thesis and the Incremental Algorithm are not concerned with the choice of value, but rather the choice of attribute, and rely on system information to guide the form the final generated expression takes, it was decided that recall was best calculated by extracting those expressions that met the constraints of the algorithms and therefore can be produced by them. That is, it was decided that recall was best calculated by analyzing each expression by hand. I will refer to this as a **per expression** analysis.

Extracting only those expressions that meet the constraints of each algorithm allows for different values of the same attribute (for example *red* or *orange* for COLOR), and provides the flexibility to account for expressions with different forms due to the selection of head noun and modifiers with differential discriminatory power (this is particularly true of the Incremental Algorithm). Minimizing bias in the analysis of each expression, either by an automated approach or by hand, requires detailed procedures for each algorithm. I will now address the reasons for this kind of analysis in detail and outline the steps taken to minimize bias in the evaluation.

Return to the example of Table 5.3. The algorithm proposed in this thesis can generate the observed expressions *orange ball* and *orange toy*, but cannot generate the expression *ball*. This is due to the different possible values for the head noun in input, and the requirement that a value for COLOR be included as long as that value is not definitive of the head noun. The proposed algorithm can also generate *red spiked ball* and *spiky red ball*,² but each of these expressions depend on different input values. To generate *orange ball*, the input referent vector must contain the head noun *ball* (as well as the attribute-value pair COLOR:‘orange’); to generate *orange toy*, the input referent vector must contain the head noun *toy*; to generate *red spiked ball*, the input referent vector must contain the head noun

²Using the type system introduced in Chapter 4, these exact orderings are predicted as well: *red* and *spiked* are both Type 6, tending to occur in position two-three, while *spiky* is Type 4, tending to occur in position four.

ball along with the attribute-value pairs COLOR:‘red’ and SHAPE:‘spiked’; to generate *spiky red ball*, the input referent vector must contain the head noun *ball* along with the attribute-value pairs COLOR:‘red’ and SHAPE:‘spiky’. Constructing input vectors in order to test whether each observed referring expression is generatable by the algorithm requires tailoring inputs to reflect my own interpretation of what each vector would have to look like to produce the observed expression. This is not only messy, but prone to the very same biases that a more automated procedure would ideally avoid.

Evaluating the Incremental Algorithm makes the reason for a per expression approach even more clear. The Incremental Algorithm cannot generate the expression *orange ball*, but can successfully generate the expressions *ball* and *orange toy*. This is due to the different possible values for preferred head noun, user knowledge, basic level values, and more specific values that are dependent on the system, not the functions of the Incremental Algorithm. If the input to the Incremental Algorithm includes information that both the ball and the surfboard should have the head noun of *toy* – for the Incremental Algorithm this head noun is derived from an input attribute-value pair TYPE:‘toy’ – the algorithm can determine that this word *toy* is the correct output and *orange* is an appropriate discriminating COLOR value by its balancing of what the system lists as more specific and known by the user with the discriminatory power of the head noun’s specificity.

There is a clear solution to this problem. Both algorithms produce expressions that obey a coherent set of rules. The way each expression is derived depends heavily on the system (particularly for the Incremental Algorithm) and the input, but the final expression itself will adhere to a very small set of core constraints. It follows that extracting those expressions that obey the coherent set of rules is an effective way of judging the recall of each algorithm. This gets rid of the complication of fabricating what the system preferences would be (a task that is inherently biased) or how the inputs would look (which would require contriving many different inputs to conform to each observed expression).

The two systems were therefore evaluated against the same set of human-generated expressions, produced for the pictures shown in Appendix A. They were judged on a per expression basis, meaning each expression was determined to be either predicted by or not

predicted by each algorithm. The two algorithms generate expressions obeying different constraints, and adherence to these constraints determined whether each expression was counted as predicted by each algorithm or not. I will now discuss the constraints for each algorithm.

The Incremental Algorithm generates referring expressions where the head noun may and each modifier moving outwards from the head noun must rule out at least one member of the contrast set. With this, each value (where *value* corresponds to the head noun and modifiers) rules out at least one member that has not yet been ruled out by an earlier value. The head noun is the only exception to this rule; it may not rule out any item in the contrast set when each item has the same head noun. Because the Incremental Algorithm finishes when all contrast items have been ruled out, the final expression must rule out all members of the contrast set. Referring expressions in the data that met this criteria were calculated as being predicted by the Incremental Algorithm. A listing of these constraints is given in Table 5.1.

The proposed algorithm generates referring expressions that have three or fewer modifiers attached to the head noun. Those closer to the head noun convey absolute properties, while those farther from the head noun convey gradable properties, here instantiated as SIZE values. There is a separation between these two kinds of values, such that there is no absolute modifier farther from the head noun than any gradable modifier. Further, the only gradable modifiers allowed are those that can be derived from comparison with an item of the same type (volume-type or surface-type) in the contrast set (including the common-value form of the intended referent). Where the contrast set contains an item that can be referred to with the same head noun as the intended referent, but differs in size, a size adjective in comparative form is allowed. Referring expressions in the data that met this criteria were calculated as being predicted by the proposed algorithm. A listing of these constraints is given in Table 5.2.

For the proposed algorithm, analyzing the data in this way provides for different possible inputs and common values, as deciding what the input is and what is common is beyond the scope of the algorithm and determined or provided by the system. For the Incremental Al-

Table 5.1: Incremental Algorithm Constraints Table 5.2: Proposed Algorithm Constraints

- | | |
|--|--|
| <ul style="list-style-type: none"> • Head noun may rule out at least one member of the contrast set. • Each modifier moving outwards from the head noun rules out at least one member of the contrast set. • No modifier only rules out members of the contrast set that are already ruled out by the head noun or modifiers closer to the head noun. • The modifiers and head noun of the full referring expression together form a uniquely distinguishing expression, ruling out all members of the contrast set. | <ul style="list-style-type: none"> • No more than three prenominal modifiers. • No absolute modifier is farther from the head noun than any gradable modifier. • Each gradable modifier is derivable from comparison with an item in the contrast set that is of the same type. • If there is a gradable modifier in comparative form, the head noun of the intended referent must also apply to an item in the contrast set from which the gradable modifier is derivable. • A color modifier is included unless it is definitive of the head noun. • A gradable modifier is included if it is derivable and the phrase contains no color modifier or the phrase contains more than one modifier. |
|--|--|

Table 5.3: Natural Variation in Referring Expression Choice

Referring Expression	Count	Referring Expression	Count
red ball	13	orange ball	12
ball	6	orange bumpy ball	3
red spiked ball	2	squishy ball	2
red dimpled ball	1	spiky red ball	2
spiked texture orange ball	1	rubber ball	1
red spikey ball	1	bumpy red ball	1
knobbly ball	1	knobby orange ball	1
rubber bumpy ball	1	spiked orange ball	1
orange plastic dryer ball	1	red sticky ball	1
round red ball	1	orange rubber ball	1
red spike ball	1	bumpy ball	1
orange nubby ball	1	orange dryer ball	1
bright red ball	1	spiky ball	1
stress ball	1	dog toy ball	1
orange dog toy	2	toy	1
orange toy	1	orange pet toy	1
orange globe	1	dog toy	1

gorithm, this allows for a variety of different results from `BasicLevelValue`, `MoreSpecificValue`, `UserKnows`, and the input `PreferredAttributes` (see Figure 2.2). This flexibility is crucial, as it allows the generation capabilities of the algorithms to be calculated in isolation from the decisions outside of their scope.

5.5 Results

5.5.1 Recall

The token recall score for the algorithm is 71.26%, capable of generating 828 of the 1,162 observed referring expressions. Running the absolute modifiers through the type system introduced in Chapter 4, where out-of-vocabulary items are predicted to be Type 8, lowers this number to 783 (a token recall of 67.38%). The Incremental Algorithm received a recall score of just 53.70%.

Table 5.4: Algorithm Results

	Proposed Algorithm	Incremental Algorithm
Recall	71.26%	53.70%

The algorithm proposed in this thesis is capable of generating the majority of distinguishing reference found. Such a high recall, given the range of objects presented and referring expressions elicited from people, shows that this algorithm can generate natural reference. If embedded in a generation system, it will successfully aid in creating natural sounding language.

The fact that the recall of the proposed algorithm is higher than the Incremental Algorithm means that it performs better *at generating natural expressions*. The Incremental Algorithm, of course, was not designed explicitly to generate natural reference, but to uniquely identify referents. The observed difference in the recall of the two algorithms is therefore expected.

These results are promising, but may be improved. The data do show some consistent expressions that were not predicted. I now turn to a discussion of these expressions and how the algorithm can be enhanced.

5.6 Discussion

As discussed above, users of the Mechanical Turk program are anonymous. Therefore, one drawback of this approach is that I have very little information about the participants, and so may have collected data from non-native speakers as well as native speakers. However, this does not appear to have created any problems.

5.6.1 Error Analysis

The expressions not predicted by the algorithm follow several clear patterns. They can be classified as not including a COLOR value; containing more than one modifier but not including a SIZE value; and exhibiting an ordering where an absolute modifier appears

farther from the head noun than a gradable modifier. I will now discuss each of these errors in detail.

The algorithm mandates that COLOR be included in all expressions, as long as the color is not a common-value for the head noun, and yet, many referring expressions did not include a value for COLOR at all. This was particularly true for the generation of reference to items with patterns of color variation. For example, many people referred to a checkered black-and-white cap (shown in cell 8 of Appendix A) as *checkered*, with no given color value. Others referred to it just with the head noun. Similarly, a striped gold-and-silver ribbon (shown in cell 6 of Appendix A) was often referred to as *striped*, without a color specification. Removing just these two referents increases the recall by over 6%. Solving how to handle the interplay between color variation, patterns, and plain colors would greatly add to the abilities of this algorithm.

Another common phenomenon not predicted by the algorithm was the lack of a value for SIZE when more than one modifier appeared. For example, *shiny silver lantern* and *lavender rubber boots* were phrases observed in the data. The algorithm predicts that if two modifiers appear, one of them will be a SIZE value. It may be argued that *shiny silver* is itself a COLOR value, and *rubber boots* is a compound noun that is itself the head noun, but I did not make these kinds of assumptions in calculating recall.

The third phenomenon not predicted by the algorithm was the appearance of modifier orderings that run contrary to Behaghel's law. For example, *blue tall trophy* and *blue long surfboard* were observed. I am not sure how best to account for this kind of phenomena, but assume that the relatively low frequency of these kinds of orderings merits treating them as outliers. These kinds of responses may be a result of the possible inclusion of non-native speakers, however, similar switching of modifiers has been reported in many other studies (cf. Danks and Glucksberg, 1971; Pechmann, 1989).

5.6.2 Extensions to the Algorithm

Further work on this algorithm involves solving how common values of referents can best be incorporated. A knowledge base drawing from Prototype Theory (Rosch et al., 1976),

where descriptors for a referent are drawn from class membership of that referent to a fuzzy set (see Lakoff, 1973 for more details on this approach), may aid in improving the algorithm in this way.

This algorithm is also only designed to generate reference to objects presented visually. To extend this approach to other kinds of reference, more work needs to be done to determine the properties involved in each. For example, to extend this algorithm to a non-visual presentation task, the form of the gradable properties used would need to be addressed (for the generation of adjectives such as *heavy* or *soft*). To extend this algorithm to reference to people, values for hair and eye color may play a more prominent role than COLOR itself (unless the person is green). This may motivate a slightly different approach for distinguishing reference to people, although the basic structure of the algorithm could remain the same.

Chapter 6

CONCLUSIONS

This thesis has explored generating two kinds of reference: distinguishing reference and describing reference. The approach developed for generating distinguishing reference is one that draws heavily from work in psycholinguistic research. The algorithm developed for the generation of distinguishing reference appears to produce results comparable to those produced by humans. The modifier type system introduced for surface realization of both kinds of reference provides a way to order modifiers prenominal, and so generate natural reference for any noun phrase preceded by modifiers.

This work has shown that there are two main kinds of properties that people use to identify items. Absolute properties tend to be included regardless of their power to uniquely identify a referent, while gradable properties are derived from comparison processes with other items in the contrast set. If a scene is parsed with a separation between the two properties, the differential treatment of each helps guide the generation of natural sounding referring expressions.

To order the modifiers derived from these properties, using a modifier type system has been shown to be extremely effective. Defining general modifier types, and deriving prenominal modifier ordering from these types, can result in well-formed, natural expressions. This allows a greater level of automation in natural language generation, where modifiers no longer must be individually ordered. Instead, by incorporating a knowledge base of modifier types, individual modifier ordering can be decided by the system.

Further work should focus on refining the algorithm, and expanding it to include generation capabilities for all kinds of reference. In particular, exploring the factors at play in the representation of the contrast set would strengthen the algorithm's ability to generate the gradable modifiers identical to those produced by people. Incorporating domain knowledge into the algorithm would further add to its ability to generate natural utterances for

a variety of tasks.

The main idea that has emerged from this study is that current incremental approaches to generating natural language do not quite capture naturalness. The system proposed here draws on evidence from a variety of sources and is capable of generating language that is quite similar to that produced by humans. Other algorithms may work best for goal-driven tasks, or be more computationally efficient. However, for the task of generating natural language, the approach introduced here is shown to be extremely effective.

BIBLIOGRAPHY

- Amazon (2008). Amazon Mechanical Turk: Artificial Artificial Intelligence.
- Antes, J. R. (1974). The Time Course of Picture Viewing. *Journal of Experimental Psychology*, 103:62–70.
- Appelt, D. E. (1981). *Planning Natural Language Utterances to Satisfy Multiple Goals*. PhD thesis, Stanford University.
- Appelt, D. E. (1985). Planning English Referring Expressions. *Artificial Intelligence*, 26:1–33.
- Areces, C., Koller, A., and Striegnitz, K. (2008). Referring Expressions as Formulas of Description Logic. *Proceedings of Fifth International Natural Language Generation Conference*, pages 42–29.
- Babcock, J. S., Lipps, M., and Pelz, J. B. (2002). How People Look at Pictures Before, During, and After Scene Capture: Buswell Revisited. In Rogowitz, B. E. and Pappas, T. N., editors, *Proceedings of SPIE Human Vision and Electronic Imaging VII*, volume 4662, pages 34–47.
- Bateman, J. A., Henschel, R., and Rinaldi, F. (1995). Generalized Upper Model 2.0: Documentation. Technical report, GMD/Institut Für Integrierte Publikations Und Informationssysteme, Darmstadt, Germany.
- Behaghel, O. (1930). *Von Deutscher Wortstellung*, volume 44. Zeitschrift Für Deutschen, Unterricht.
- Bever, T. G. (1970). The Cognitive Basis for Linguistic Structures. In Hayes, J. R., editor, *Cognition and the Development of Language*. Wiley, New York.
- Boleda, G. and Alonso, L. (2003). Clustering Adjectives for Class Acquisition. In *Proceedings of the EACL’03 Student Session*, pages 9–16, Budapest.
- Boleda, G., Badia, T., and Schulte Im Walde, S. (2005). Morphology Vs. Syntax in Adjective Class Acquisition. In *Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*, pages 77–86, Ann Arbor, Michigan.
- Brants, T. and Franz, A. (2006). Web 1T 5-gram Version 1.
- Bresnan, J. (2001). *Lexical-Functional Syntax*. Blackwell Publishers, Oxford.
- Carroll, J., Copestake, A., Flickinger, D., and Poznanski, V. (1999). An Efficient Chart Generator for (Semi-)Lexicalist Grammars. *Proceedings of the 7th European Workshop on Natural Language Generation*, pages 86–95.

- Clark, H. H. and Clark, E. V. (1977). *Psychology and Language*. Harcourt Brace Jovanovich, New York.
- Clark, H. H. and Wilkes Gibbs, D. (1986). Referring as a Collaborative Process. *Cognition*, 22:1–39.
- Cooper, W. E. and Ross, J. R. (1975). Word Order. *Papers on the Parasession on Functionalism*, pages 63–111.
- Dale, R. (1988). *The Generation of Subsequent Referring Expressions in Structured Discourses*, pages 58–75. Pinter Publishers, London.
- Dale, R. (1989). Cooking up Referring Expressions. In *Proceedings of the Twenty-Seventh Annual Meeting of the Association for Computational Linguistics*, Vancouver, British Columbia.
- Dale, R. (1992). *Generating Referring Expressions: Building Descriptions in a Domain of Objects and Processes*. MIT Press, Cambridge.
- Dale, R. and Haddock, N. (1991). Content Determination in the Generation of Referring Expressions. *Computational Intelligence*, 7:252–265.
- Dale, R. and Reiter, E. (1995). Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science*, 18:233–263.
- Danks, J. H. (1976). Language-Processing Mechanisms Are Not Unique: a Reply to Richards' Critique. *The American Journal of Psychology*, 89(1):137–145.
- Danks, J. H. and Glucksberg, S. (1971). Psychological Scaling of Adjective Order. *Journal of Verbal Learning and Verbal Behavior*, 10:63–67.
- Danks, J. H. and Swenk, M. A. (1972). Prenominal Adjective Order and Communication Context. *Journal of Verbal Learning and Verbal Behavior*, 11:183–187.
- Danks, J. H. and Swenk, M. A. (1974). Comprehension of Prenominal Adjective Order. *Memory and Cognition*, 2:34–38.
- Dell, G. (1985). Positive Feedback in Hierarchical Connectionist Models: Applications to Language Production. *Cognitive Science*, 9:3–24.
- Donnellan, K. (1966). Reference and Definite Description. *Philosophical Review*, 75:281–304.
- Gardent, C. (2002). Generating Minimal Definite Descriptions. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 96–103.
- Gardent, C., Manuélian, H., Striegnitz, K., and Amoia, M. (2004). Generating Definite Descriptions: Nonincrementality, Inference, and Data. In *Multidisciplinary Approaches to Language Production*. Mouton De Gruyter.

- Gatt, A. (2008). TUNA: Towards a Unified Algorithm for the Generation of Referring Expressions.
- Gatt, A. and Belz, A. (2008). Attribute Selection for Referring Expression Generation: New Algorithms and Evaluation Methods. *Proceedings of Fifth International Natural Language Generation Conference*, pages 50–58.
- Greenberg, J. H. (1963). *Some Universals of Grammar with Particular Reference to the Order of Meaningful Elements*, pages 73–113. MIT Press, London.
- Grice, P. H. (1975). Logic and Conversation. *Syntax and Semantics*, 3:41–58.
- Griffin, Z. M. and Bock, K. (2000). What the Eyes Say About Speaking. *Psychological Science*, 11:274–279.
- Grosz, B. J. (1977). *The Representation and Use of Focus in Dialogue Understanding*. PhD thesis, University of California, Berkeley.
- Halliday, M. and Matthiessen, C. (1999). *Construing Experience As Meaning: a Language-Based Approach to Cognition*. Cassell.
- Hermann, T. and Deutsch, W. (1976). *Psychologie Der Objektbenennung*. Huber Verlag, Bern.
- Herring, J. R. (1990). The Mathematical Modeling of Spatial and Non-Spatial Information in Geographic Information Systems. *Proceedings of the NATO Advanced Study Institute on Cognitive and Linguistic Aspects of Geographic Space*.
- Horacek, H. (1997). An Algorithm for Generating Referential Descriptions with Flexible Interfaces. In Cohen, P. R. and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, Somerset, New Jersey.
- Jupiterimages Corporation (2008). <http://www.clipart.com>.
- Kempen, G. and Hoenkamp, E. (1987). An Incremental Procedural Grammar for Sentence Production. *Cognitive Science*, 11:201–258.
- Krahmer, E. and Theune, M. (2002). Efficient Context-Sensitive Generation of Descriptions. In van Deemter, K. and Kibble, R., editors, *Information Sharing: Givenness and Newness in Language Processing*, pages 223–264, Stanford, California. CSLI Publications.
- Krahmer, E., van Erk, S., and Verleg, A. (2003). Graph-Based Generation of Referring Expressions. *Computational Linguistics*, 29(1):53–72.
- Kronfeld, A. (1986). Donnellan’s Distinction and a Computational Model of Reference. *Proceedings of the Twenty-Fourth Annual Meeting of Association for Computational Linguistics*, pages 186–191.

- Kronfeld, A. (1989). Conversationally Relevant Descriptions. *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*.
- Lakoff, G. (1973). Hedges: a Study in Meaning Criteria and the Logic of Fuzzy Concepts. *Journal of Philosophical Logic*, 2:458–508.
- Landau, B. and Jackendoff, R. (1993). “What” and “Where” in Spatial Language and Spatial Cognition. *Behavioral and Brain Sciences*, 16:217–265.
- Levelt, W. (1989). *Speaking: from Intention to Articulation*. MIT Press.
- Loftus, G. R. and Mackworth, N. H. (1978). Cognitive Determinants of Fixation Location During Picture Viewing. *Journal of Experimental Psychology: Human Perception and Performance*, 4(4):565–572.
- MacWhinney, B. (1982). Basic Syntactic Processes. In Kuczaj, S., editor, *Language Development: Syntax and Semantics*, volume 1, pages 72–136, Hillsdale, NJ. Lawrence Erlbaum.
- Malouf, R. (2000). The Order of Prenominal Adjectives in Natural Language Generation. In *Proceedings of 38th Annual Meeting of the Association for Computational Linguistics*, pages 85–92, Hong Kong.
- Manning, C. D. (1993). Automatic Acquisition of a Large Subcategorization Dictionary from Corpora. In *Meeting of the Association for Computational Linguistics*, pages 235–242.
- Martin, J. E. (1969a). Semantic Determinants of Preferred Adjective Order. *Journal of Verbal Learning and Verbal Behavior*, 8:697–704.
- Martin, J. E. (1969b). Some Competence-Process Relationships in Noun Phrases with Prenominal and Postnominal Adjectives. *Journal of Verbal Learning and Verbal Behavior*, 8:471–480.
- McDonald, D. D. (1980). *Natural Language Generation as a Process of Decision-Making Under Constraints*. PhD thesis, Massachusetts Institute of Technology.
- McKeown, K. (1985). *Text Generation*. Cambridge University Press, New York, NY.
- Miller, G. A. (2006). WordNet: a Lexical Database for the English Language.
- Oberlander, J. and Dale, R. (1991). Generating Expressions Referring to Eventualities. *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, pages 67–72.
- O’Donnell, M. (1996). Input Specification in the WAG Sentence Generation System. *Proceedings of the Eighth International Workshop on Natural Language Generation*.
- O’Donnell, M. (2000). Museum Audio Guides Which Adapt to the User and the Context. *Proceedings of the First Jornadas en Tecnología del Habla*.

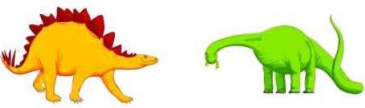





- Passonneau, R. (2006). Measuring Agreement on Set-Valued Items (MASI) for Semantic and Pragmatic Annotation. *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 831–836.
- Pechmann, T. (1989). Incremental Speech Production and Referential Overspecification. *Linguistics*, 27:89–110.
- Ratnaparkhi, A. (1997). A Simple Introduction to Maximum Entropy Models for Natural Language Processing. *Technical Report 97-08*.
- Rayner, K. and Pollatsek, A. (1992). Eye Movements and Scene Perception. *Canadian Journal of Psychology*, 46:342–376.
- Reiter, E. (1990a). The Computational Complexity of Avoiding Conversational Implicatures. In *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics*, pages 97–104, Morristown, NJ, USA. Association for Computational Linguistics.
- Reiter, E. (1990b). The Computational Complexity of Avoiding False Implicatures. *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*.
- Reiter, E. (1990c). Generating Descriptions That Exploit a User’s Domain Knowledge. In Mellish, C. and Zock, M., editors, *Current Research in Natural Language Generation*, pages 257–285. Academic Press Professional, Inc., San Diego, CA, USA.
- Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press.
- Richards, M. M. (1975). The Pragmatic Communication Rule of Adjective Ordering: a Critique. *The American Journal of Psychology*, 88(2):201–215.
- Rosch, E., Mervis, C. B., Gray, W., Johnson, D., and Boyes Braem, P. (1976). Basic Objects in Natural Categories. *Cognitive Psychology*, 8(3):382–439.
- Sag, I. A., Wasow, T., and Bender, E. M. (2003). *Syntactic Theory: a Formal Introduction*. CSLI Publications, Stanford University, California.
- Shaw, J. and Hatzivassiloglou, V. (1999). Ordering Among Premodifiers. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 135–143, Morristown, NJ, USA. Association for Computational Linguistics.
- Sinclair, J. (1992). Trust the Text. In Davies, M. and Ravelli, L., editors, *Advances in Systemic Linguistics*, pages 5–19. Pinter, London.
- Sonnenschein, S. (1985). The Development of Referential Communication Skills: Some Situations in Which Speakers Give Redundant Messages. *Journal of Psycholinguistic Research*, 14:489–508.




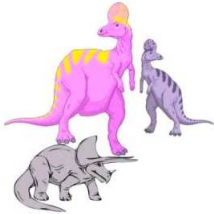


- Sripada, S. G., Reiter, E., and Dale, R. (2003). Generating English Summaries of Time Series Data Using the Gricean Maxims. In *KDD03*, pages 187–196.
- Stone, M. (2000). On Identifying Sets. In *In Proceedings of INLG-2000, Mitzpe*, pages 116–123.
- Stone, M. and Webber, B. (1998). Textual Economy Through Close Coupling of Syntax and Semantics. In Hovy, E., editor, *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 178–187. Association for Computational Linguistics, New Brunswick, New Jersey.
- Strawson, P. F. (1956). On Referring. *Mind*, 59:320–344.
- Stubbs, M. (2001). *Words and Phrases: Corpus Studies of Lexical Semantics*. Blackwell, Cornwall.
- Toutanova, K., Klein, D., Manning, C., and Singer, Y. (2003). Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. *Proceedings of HLT-NAACL 2003*, pages 252–259.
- Toutanova, K. and Manning, C. D. (2000). Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, Hong Kong.
- van Deemter, K. (2000). Generating Vague Descriptions. In *Proceedings of the First Natural Language Generation Conference*, pages 12–16.
- van Deemter, K. (2002). Generating Referring Expressions: Boolean Extensions of the Incremental Algorithm. *Computational Linguistics*, 28(1):37–52.
- van Deemter, K. (2004). Generating Referring Expressions That Involve Gradable Properties. *Computational Linguistics*, 32(2):195–222.
- van der Meulen, F. F., Meyer, A. S., and Levelt, W. J. M. (2001). Eye Movements During the Production of Nouns and Pronouns. *Memory and Cognition*, 29:512–521.
- van Der Sluis, I. and Krahmer, E. (2004). Evaluating Multimodal NLG Using Production Experiments. *Proceedings of the 4th LREC*, pages 209–212.
- Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.
- Vendler, Z. (1968). *Adjectives and Nominalizations*. Mouton.
- Viethen, J. and Dale, R. (2006). Algorithms for Generating Referring Expressions: Do They Do What People Do? *Proceedings of the Fourth International Natural Language Generation Conference*, pages 63–70.

- Viethen, J. and Dale, R. (2008). The Use of Spatial Relations in Referring Expression Generation. *Proceedings of Fifth International Natural Language Generation Conference*, pages 59–67.
- von Kleist, H. (1961). Über Die Allmähliche Verfertigung Der Gedanken Beim Reden. In Sembdner, H., editor, *From Sämtliche Werke Und Briefe*. Wissenschaftliche Buchgesellschaft, Darmstadt.
- Whorf, B. L. (1945). Grammatical Categories. *Language*, 21(1):1–11.
- Winograd, T. (1972). *Understanding Natural Language*. Academic Press.
- Wulff, S. (2003). A Multifactorial Corpus Analysis of Adjective Order in English. *International Journal of Corpus Linguistics*, 8:245–282.
- Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, 8:338–3.
- Ziff, P. (1960). *Semantic Analysis*. Cornell University Press, Ithaca, New York.
- Zock, M., Sabah, G., and Alviset, C. (1986). From Structure to Process. Computer-Assisted Teaching of Various Strategies for Generating Pronoun Constructions in French. In *COLING*, pages 566–569.

Appendix A

PICTURES FROM EVALUATION

<p style="text-align: center;">1</p>  <p>_____ is smiling at _____.</p>	<p style="text-align: center;">2</p>  <p>_____ is to the left of _____.</p>
<p style="text-align: center;">3</p>  <p>_____ is carrying _____.</p>	<p style="text-align: center;">4</p>  <p>_____ is on top of _____.</p>
<p style="text-align: center;">5</p>  <p>_____ is to the left of _____.</p>	<p style="text-align: center;">6</p>  <p>_____ on the left is tied with _____.</p>
<i>Continued on Next Page</i>	

<i>Continued from Previous Page</i>	
<p>7</p>  <p>_____ has _____ and _____ on top of it.</p>	<p>8</p>  <p>_____ is carrying _____ and wearing _____ on her head.</p>
<p>9</p>  <p>_____ are between _____.</p>	<p>10</p>  <p>_____ is standing on top of _____.</p>
<p>11</p>  <p>_____ is wearing _____ and playing _____.</p>	<p>12</p>  <p>_____ is on the far left side and _____ is on the far right side.</p>
<p><i>Continued on Next Page</i></p>	

Continued from Previous Page

13



_____ is in the middle.

14



_____ is petting _____.

15



_____ is on the left side and _____ is on the right side.

16



_____ is wearing _____.

17



_____ is twirling _____.

18



_____ is on the right side.

19



_____ on the left is wearing _____ on her head.

20



_____ is hiding behind _____.

Appendix B

FILTERED MODIFIERS FOR THE MODIFIER CLASSIFICATION SYSTEM

Table B.1: Filtered Modifiers in Modifier Type System

a	b	c	d	e	f
g	h	i	j	k	l
m	n	o	p	q	r
s	t	u	v	w	x
y	z	in	on	up	above
before	after	during	most	more	much
each	every	all	some	few	any
only	many	other	very	such	well
no	pussy	like	said	just	cd
cl	il	cc	cd		

Appendix C

CONTRAST SET MAPPINGS

Table C.1: Example Contrast Set Properties: Size Relations and Corresponding Adjectives

	Contains Relations	Generates
	taller_than longer_than wider_than shorter_than thinner_than less_wide_than taller_than thinner_than longer_than thinner_than longer_than less_wide_than taller_than less_wide_than	tall long wide fat short thin thin tall thin long thin long thin tall thin
	taller_than longer_than taller_than wider_than longer_than wider_than shorter_than thinner_than shorter_than less_wide_than thinner_than less_wide_than taller_than longer_than wider_than thinner_than less_wide_than shorter_than	large big large big large big small little small little small little large big small little
Surface-type mappings	taller_than longer_than wider_than	large big large big large big

Appendix D

MECHANICAL TURK

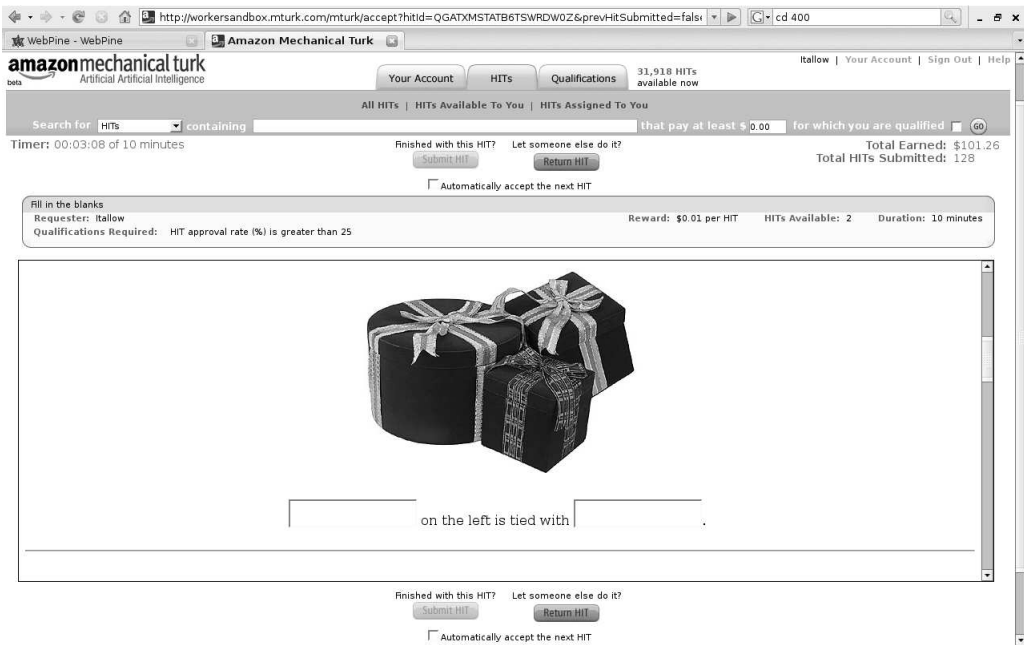


Figure D.1: Mechanical Turk: Screenshot