Class-Based Ordering of Prenominal Modifiers

Margaret Mitchell
Center for Spoken Language Understanding
Portland, Oregon, U.S.A
itallow@cslu.ogi.edu

Abstract

This paper introduces a class-based approach to ordering prenominal modifiers. Modifiers are grouped into broad classes based on where they tend to occur prenominally, and a framework is developed to order sets of modifiers based on their classes. This system is developed to generate several orderings for modifiers with more flexible positional constraints, and lends itself to bootstrapping for the classification of previously unseen modifiers.

1 Introduction

Ordering prenominal modifiers is a necessary task in the generation of natural language. For a system to effectively generate fluent utterances, the system must determine the proper order for any given set of modifiers. The order of modifiers before a noun affects the meaning and fluency of generated utterances. Determining ways to order modifiers prenominally has been an area of considerable research (cf. Shaw and Hatzivassiloglou, 1999; Malouf, 2000).

In this paper, we establish and evaluate a classification system that can be used to order prenominal modifiers automatically. This may be implemented in a surface realization component of a natural language generation system, or may be used to help specify the ordering of properties that feed into a referring expression generation algorithm. Predictions of prenominal modifier ordering based on these classes are shown to be robust and accurate.

The work here diverges from the approaches commonly employed in modifier classification by assuming no underlying relationship between semantics and prenominal order or morphology and prenominal order. The approach instead relies on generalizing empirical evidence from a corpus. This allows the system to be robust and portable to a variety of applications, without precluding any underlying linguistic constraints.

In the next section, we discuss prior work on this topic, and address the differences in our approach. Section 3 discusses the relationship between modifier ordering and referring expression generation, a principal component of natural language generation. Section 4 describes the ideas behind the modifier classification system. Sections 5 and 6 present the materials and methodology of the current study, with a discussion of the corpus involved and the basic modules used in the process. In Section 7 we discuss the results of our study. Finally, in Section 8, we outline areas for improvement and possible future work.

2 Related Work

Discerning the rules governing the ordering of adjectives has been an area of research for quite some time (see, for example, Panini's work on Sanskrit grammar ca. 350 BCE). Most approaches assume an underlying relationship between semantics and prenominal position (cf. Whorf, 1945; Ziff, 1960; Bever, 1970; Danks and Glucksberg, 1971). 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 in a generation system, as they require determining the semantic properties of each modifier used, relative to the context in which it occurs. If a modifier classification

scheme is to be implemented, it should be able to create a variety of natural, unmarked orders; be robust enough to handle a wide variety of modifiers; and be flexible enough to allow different natural orderings.

Shaw and Hatzivassiloglou (1999) examine this problem, and develop ways to order all prenominal modifier types. This includes adjectives as well as nouns, such as "baseball" in "baseball field"; gerunds, such as "running" in "running man"; and past participles, such as "heated" in "heated debate". The authors devise three different methods that may be implemented in a generation system to order these kinds of prenominal modifiers. These are the *direct evidence* method, the *transitivity* method, and the *clustering* method.

Briefly, given prenominal modifiers *a* and *b* in a training corpus, the direct evidence method utilizes probabilistic reasoning to determine whether the frequency count of the ordered sequence <*a*,*b*> or <*b*,*a*> is stronger. The transitivity method makes inferences about unseen orderings among prenominal modifiers; given a third prenominal modifier *c*, where *a* precedes *b* and *b* precedes *c*, the authors can conclude that *a* precedes *c*. In the clustering method, an order similarity metric is used to group modifiers together that share a similar relative order to other modifiers.

Shaw and Hatzivassiloglou achieve their highest prediction accuracy of 90.67% using their transitivity technique on prenominal modifiers from a medical corpus. However, with their system trained on the medical corpus and then tested on the Wall Street Journal corpus (Marcus et al., 1993), they achieve an overall prediction accuracy of only 54%. The authors conclude that prenominal modifier ordering is domain-specific.

Malouf (2000) continues this work, determining the order for sequences of prenominal adjectives by examining several different statistical and machine learning techniques. These achieve good results, ranging from 78.28% to 89.73% accuracy. Malouf achieves the best results by combining memory-based learning and positional probability, which reaches 91.85% accuracy at predicting the prenominal adjective orderings in the first 100 million tokens of the BNC. However, this analysis does not extend to other kinds of prenominal modifiers. The model is also not tested on a different domain.

The approach to modifier classification taken here is similar to the clustering method discussed by Shaw and Hatzivassiloglou. Modifiers are grouped into classes based on where they occur prenominally. This approach differs, however, in how classes are assigned. In our approach, modifiers are grouped into classes based on the frequencies with which they occur in different prenominal positions. Classes are built based not on where modifiers are positioned in respect to other modifiers, but on where modifiers are positioned in general. Grouping modifiers into classes based on prenominal positions mitigates the problems noted by Shaw and Hatzivassiloglou that ordering predictions cannot be made (1) when both *a* and *b* belong to the same class, (2) when either *a* or *b* are not associated to a class that can be ordered with respect to the other, and (3) when the evidence for one class preceding the other is equally strong for both classes.

This approach allows modifiers with strong positional preferences to be in a class separate from modifiers with weaker positional preferences. This also ensures that any prenominal modifiers *a* and *b* seen in the training corpus can be ordered, regardless of which particular modifiers they appear with and whether they occur together in the training data at all. This approach also has the added benefit of developing modifier classes that are usable across many different domains. Further, this method is conceptually simple and easy to implement. Although this approach is less context-sensitive than earlier work, we find that it is highly accurate, with comparable token precision. We discuss this in greater detail in Sections 6 and 7.

3 The Problem of Ordering Prenominal Modifiers

Generating referring expressions in part requires generating the adjectives, verbs, and nouns that modify head nouns. In order for these expressions to clearly convey the intended referent, the modifiers must appear in an order that sounds natural and mimics human language use.

For example, consider the alternation given in Figure 1. Some combinations of modifiers before a noun are more marked than others, although all are strictly speaking grammatical. This speaks to the need for a broad modifier classes to order prenominal modifiers, where individual modifiers

(1) big beautiful white wooden house

(2) ?white wooden beautiful big house

(3) comfortable red chair

(4) ?red comfortable chair

(5) big rectangular green Chinese silk carpet

(6) ?Chinese big silk green rectangular carpet

Figure 1: Grammatical Modifier Alternations (Vendler, 1968: 122)

may be ordered separately as required by particular contexts.

Along these lines, almost all referring expression generation algorithms rely on the availability of a predefined ordering or weighting of properties (Dale and Reiter, 1995; van Deemter, 2002; Krahmer et al., 2003). This requires that for every referent, an ordered or weighted listing of all the properties that can apply to it must be created before referring expression generation begins. In these models, the order or weights of the input properties map to the order of the output modifiers.

However, the method used to determine the ordering or weighting of properties is an open issue. The difficulty with capturing the ordering of properties and their corresponding modifiers stems from the problem of data sparsity. In the example in Figure 1, 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 sequence is small. The data is just too sparse.

4 Towards a Solution

To create a flexible system capable of predicting a wide variety of orderings, we used several corpora to build broad modifier classes. Modifiers are classified by where they tend to appear prenominally, and ordering constraints between the classes determine the order for any set of modifiers. This system incorporates three main ideas:

1. Not all modifiers have equally stringent ordering preferences.
2. Modifier ordering preferences can be learned empirically.
3. Modifiers can be grouped into classes indicative of their ordering preferences.

The classification scheme therefore 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: This classification scheme builds on what there is direct evidence for, independent of why each modifier appears where it does.

To create our model, all simplex noun phrases (NPs) are extracted from parsed corpora. A *simplex NP* is defined as a maximal noun phrase that includes premodifiers such as determiners and possessives but no post-nominal constituents such as prepositional phrases or relative clauses (Shaw and Hatzivassiloglou, 1999: 137). From these simplex NPs, we extract all those headed by a noun and preceded by only prenominal modifiers. This includes modifiers tagged as adjectives (JJ), nouns (NN), gerunds (VBG), and past participles (VBN). The counts and relative positions of each modifier are stored, and these are converted into position probabilities in vector file format. Modifiers are classified based on the positions in which they have the highest probabilities of occurring.

Examples of the intermediary files in this process are given in Tables 1 and 2. Table 1 lists modifiers followed by their frequency counts in each prenominal position. Table 2 lists these modifiers associated to their classes, with the proportions that determine the class.

wealthy	four 2	three 2	two 3	one 1
red	four 13	three 35	two 35	one 21
golden	four 1	three 5	two 5	one 3
strongest	four 5	three 5	two 5	one 5

Table 1: Example Modifier Classification Intermediate File: Step 3

5 Materials

To create the training and test data, we utilize the Penn Treebank-3 (Marcus et al., 1999) releases of

wealthy	two	two	0.38				
red	two_three	three	0.34	two	0.34		
golden	one_two_three	three	0.33	two	0.33	one	0.29
strongest	two_three_four	four	0.33	three	0.33	two	0.33

Table 2: Example Modifier Classification Intermediate File: Step 4

the parsed Wall Street Journal corpus, the parsed Brown corpus, and the parsed Switchboard corpus. The Wall Street Journal corpus is a selection of over one million words collected from the Wall Street Journal over a three-year period. The Brown corpus is over one million words of prose written in various genres, including mystery, humor, and popular lore, collected from newspapers and periodicals in 1961. The Switchboard corpus is over one million words of spontaneous speech collected from thousands of five-minute telephone conversations. Several programs were constructed to analyze the information provided by these data. The details of each module are outlined below.

5.1 Code Modules

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

Modifier Extractor – This program takes as input a parsed corpus, and outputs a list of all occurrences of all noun phrases in that corpus.
input: Parsed Corpus
output: List of simplex NPs

Modifier Organizer – This program takes as input a list of simplex NPs and filters out words that appear prenominally and are occasionally mistagged as modifiers. A list of these filtered words is available in Table 3. 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 builds our model by determining the classification for each modifier.

about	behind	on
above	in	under
after	inside	out
outside	up	over
down	like	past
near	through	off
the	a	

Table 3: Filtered Mistagged Words

input: Vector file with distributional information for each modifier position
output: Ordering model: File with each modifier associated to a class

Prenominal Modifier Ordering Predictor – This program takes as input two files: an ordering model and a list of simplex NPs (for testing). The program then uses the model to assign 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 decisions 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 simplex NPs
output: Precision and recall for modifier ordering predictions

6 Method

6.1 Classification Scheme

To develop modifier classes and create our model, we assume four primary modifier positions. This assumption is based on the idea that people rarely produce more than four modifiers before a noun. This assumption covers 99.70% of our data (see Table 5). The longest noun phrases for this experiment are therefore those with five words: Four modifiers followed by a noun.

small	smiling	white	fuzzy	bunny
four	three	two	one	

Figure 2: Example Simplex NP with Prenominal Positions

Each modifier's class is determined by counting the frequency of each modifier in each position.

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

Table 4: Modifier Classes

This is turned into a probability over all four positions. All position probabilities ≤ 0.25 (baseline) are discarded. Those positions that remain determine the modifier class.

To calculate modifier position for each phrase, counts were incremented for all feasible positions. This is a way of sharing evidence among several 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 broad 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.

The final class for each modifier is dependent on the positions the modifier appears in more than 25% of the time. Since there are four possible positions, 25% is the baseline: A single modifier preceding a noun has equal probability of being in each of the four positions. There are nine derivable modifier classes in this approach, listed in Table 4.

A diagram of how a modifier is associated to a class is shown in Figure 3. In this example, *red* appears in several simplex NPs. In each sequence, we associate *red* to its possible positions within the four prenominal slots. We see that *red* occurs in positions one, two and three; two, three, and four; and three and four. With only this data, *red* has a 12.5% probability of being in position one; a 25% probability of being in position two; a 37.5% probability of being in position three; and a 25% probability of being in position four. It can therefore be classified as belonging to Class 3, the class for modifiers that tend to occur in position three.

This kind of classification allows the system to be flexible to the idea that some modifiers exhibit stringent ordering constraints, while others have more loose constraints. Some modifiers may always appear immediately before the noun, while

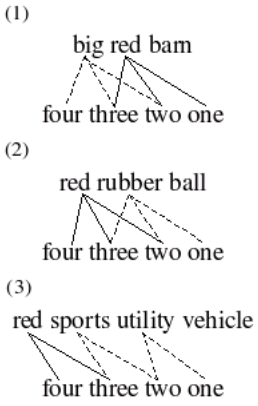

Figure 3: Constructing the Class of the Modifier *red*

others may generally appear close to or far from the noun. By counting the occurrences of each modifier in each position, classes for all observed modifiers may be derived.

The frequencies of all extracted groupings of prenominal modifiers used to build our model are shown in Table 5. The frequencies of the extracted classes are shown in Table 6.

Mods	Count	Percentage
2	15856	88.90%
3	1770	9.92%
4	155	0.87%
5	21	0.12%
6	1	.03%

Table 5: Frequency of Prenominal Modifier Amounts

Class	Position	Count	Percentage
1	one	18	0.23%
2	two	46	0.68%
3	three	62	0.92%
4	four	21	0.31%
5	one-two	329	4.88%
6	two-three	1136	16.86%
7	three-four	261	3.87%
8	one-two-three	2671	39.65%
9	two-three-four	2193	32.55%

Table 6: Modifier Class Distribution

Modifiers of Class 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. An example of a modifier from Class 8 is *golden*. The next

Class	Position	Generated Before Class							
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						

Table 7: Proposed Modifier Ordering

largest portion of the data are modifiers of Class 9, the class for modifiers that show a general preference to be farther from the head noun. An example of a modifier from Class 9 is *strongest*. With these defined, *strongest golden arch* is predicted to sound grammatical and unmarked, but *?golden strongest arch* is not.

Some expected patterns also emerge in these groupings. For example, *green*, *yellow*, *red* and other colors are determined to be Class 6. *Explained* and *unexplained* are both determined to be Class 5, and *big* and *small* are both determined to be Class 9.

Once classified, modifiers may be ordered according to their classes. The proposed ordering constraints for these classes are listed in Table 7. Note that using this classification scheme, the ordering of modifiers that belong to the same class is not predicted. This seems to be reflective of natural language use. For example, both *wealthy* and *performing* are predicted to be Class 2. This seems reasonable; whether *wealthy performing man* or *performing wealthy man* is a more natural ordering of prenominal modifiers is at least debatable. The freedom of intra-class positioning allows for some randomization in the generation of prenominal modifiers, where other factors may be used to determine the final ordering.

6.2 Evaluation

In order to test how well the proposed system works, 10-fold cross-validation was used on the extracted corpora. 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 resulting model was used to predict the prenominal ordering of modifiers in the remaining 10%.

The modifiers preceding each noun were stored in unordered groups, and the ordering for each unordered prenominal modifier pair $\{a,b\}$ was predicted based on the classes of the modifiers in our model. The ordering predictions followed the constraints listed in Table 7. When the class was known for one modifier but not for the other, the two modifiers were ordered based on the class of the known modifier: Modifiers in Classes 1, 2, 5, and 8 were placed closer to the head noun than the unknown modifier, while modifiers in Classes 3, 4, 7, and 9 were placed farther from the head noun than the unknown modifier. If the known modifier was of Class 6 (occurring in position two-three), a random guess decided the ordering. This reflects the idea that Classes 1, 2, 5, and 8 are all classes for modifiers that broadly prefer to be closer to the head noun, while Classes 3, 4, 7, and 9 are all classes for modifiers that broadly prefer to be farther from the head noun.

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 tp/(tp + fp), where tp is the number of 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 tp/(tp + 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 by our system. This measure provides information about the proportion of modifiers in the training data that can be correctly ordered.

(7) **Precision** = tp/(tp + fp)
tp = number of orderings correctly predicted
fp = number of orderings not correctly predicted

(8) **Recall** = tp/(tp + fn)
tp = number of orderings correctly predicted
fn = number of orderings that could not be predicted

	Precision	Recall
Token	89.63% (0.02)	74.14% (0.03)
Type	90.26% (0.02)	67.17% (0.03)

Table 8: Precision and Recall for Prenominal Modifier Ordering

7 Results

Results are shown in Table 8. Our model predicts the correct order for 89.63% of unordered modifiers $\{a,b\}$ for which an ordering decision can be made, making correct predictions for 74.14% of all unordered modifiers in the test data. The system also correctly predicts 90.26% of the unordered modifier $\{a,b\}$ types in the test data for which an ordering decision can be made. This covers 67.17% of the modifier pair types in the test data. This lower value appears to be due to the large amount of modifier pairs that are in the data only once.

The values given are averages over each trial. The standard deviation for each average is given in parentheses. On average, 191 modifier pairs were ordered in each trial, based on the assigned orders of 273 individual modifiers, with an average of 23.01% of the modifiers outside of the vocabulary in each trial.

The system precision and recall here are comparable to previously reported results (see Section 2). Extending our analysis over entire simplex NPs, where we generate all possible orderings based on our system constraints, we are able to predict an average of 94.44% of the sequences for which a determination can be made. This is a correct prediction for 78.59% of all the simplex NPs in the data.

Previous attempts have achieved very poor results when testing their models on a new domain. We conclude our analysis by testing the accuracy of our models on different domains. To do this, we combine two corpora to build our model and then test this model on the third.

Combining the WSJ corpus and the Brown corpus to build our modifier classes and then testing on the Switchboard (Swbd) corpus, we achieve quite promising results. Our token precision is 89.57% and our type precision is 94.17%. However, our recall values are much lower than those reported above (63.47% and 58.18%). Other training and testing combinations follow this pattern: A model built from the Switchboard corpus and the WSJ corpus achieves 82.75% token precision and 87% type precision when tested on the Brown corpus (57.14% token recall, 51.18% type recall), while a model built from the Switchboard corpus and the Brown corpus achieves 79.82% token precision and 82.43% type precision when tested on the WSJ corpus (39.55% token recall and 27.16% type recall).

Training Corpus	**Testing Corpus**	**Token Precision**	**Token Recall**
Brown+WSJ	Swbd	89.57%	63.47%
Swbd+WSJ	Brown	82.75%	57.14%
Swbd+Brown	WSJ	79.82%	39.55%
Training Corpus	**Testing Corpus**	**Type Precision**	**Type Recall**
Brown+WSJ	Swbd	94.17%	58.18%
Swbd+WSJ	Brown	87.00%	51.18%
Swbd+Brown	WSJ	82.43%	27.16%

Table 9: Precision and Recall for Prenominal Modifier Ordering of a New Domain

8 Discussion

The system precision is comparable to previously reported results. The results show that ordering modifiers based on this classification system can aid in generating simplex noun phrases with prenominal modifiers ordered in a way that sounds natural. We now turn to a discussion of areas for future work.

It seems reasonable that the classes for previously unseen modifiers could be developed based on the known classes of surrounding modifiers. This system lends itself to bootstrapping, where a lexical acquisition task that constructed class probabilities based on the surrounding context could classify previously unseen modifiers:

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.

Grouping modifiers into classes that determine their order also lends itself to incorporation into generative grammars. For example, Head-driven Phrase Structure Grammar (Sag et al., 2003), a constraint-based grammatical framework that groups lexical items into broader classes, could utilize the classes proposed here to determine modifier positions prenominally. Advancing re-

search in this area could help grow the generative capabilities of class-based grammars.

It bears mentioning that this same system was attempted on the Google Web 1T 5-Gram corpus (Brants and Franz, 2006), where we used WordNet (Miller et al., 2006) to extract sequences of nouns preceded by modifiers. The precision and recall were similar to the values reported here, however, the proportions of prenominal modifiers betrayed a problem in using such a corpus for this approach: 82.56% of our data had two prenominal modifiers, 16.79% had four, but only 0.65% had three. This pattern was due to the many extracted sequences of modifiers preceding a noun that were not actually simplex NPs. 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, but the 5-Gram itself may not be a noun phrase. We found that many of our extracted 5-Grams were actually lists of words (for example, *Chinese Polish Portuguese Romanian Russian* was observed 115 times). In the future, we would like to examine ways to use the 5-Gram corpus to supplement our system.

The results reported here are encouraging, and we hope to continue this work on a parsed version of the Gutenberg corpus (Hart, 2009). This corpus is a collection of text versions of novels and other written works, and is available online. Using a corpus of modifier-rich text such as this would aid the system in classifying a greater number of modifiers. Further work should also test how robust the acquisition of unseen modifiers is using these classes, and examine implementing this ordering system into a language generation system.

References

Otto Behaghel. 1930. *Von Deutscher Wortstellung*, volume 44. Zeitschrift Für Deutschen, Unterricht.

Thomas G. Bever. 1970. The cognitive basis for linguistic structures. In J. R. Hayes, editor, *Cognition and the Development of Language*. Wiley, New York.

Gemma Boleda and Laura Alonso. 2003. Clustering adjectives for class acquisition. In *Proceedings of the EACL'03 Student Session*, pages 9–16, Budapest.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. http://www.ldc.upenn.edu. Linguistic Data Consortium.

H. H. Clark and E. V. Clark. 1976. *Psychology and language: An introduction to psycholinguistics.* Harcourt Brace Jovanovich, New York.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 18:233–263.

M.A.K. Halliday and Christian Matthiessen. 1999. *Construing experience as meaning: a language-based approach to cognition*. Cassell, London.

Michael Hart. 2009. Project Gutenberg collection. http://www.gutenberg.org. Project Gutenberg.

Emiel Krahmer, Sebastiaan van Erk, and André Verleg. 2003. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72.

Robert Malouf. 2000. The order of prenominal adjectives in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 85–92, Hong Kong.

Christopher D. Manning. 1993. Automatic acquisition of a large subcategorization dictionary from corpora. In *Meeting of the Association for Computational Linguistics*, pages 235–242.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3. Linguistic Data Consortium.

George A. Miller, Christiane Fellbaum, Randee Tengi, Pamela Wakefield, Helen Langone, and Benjamin R. Haskell. 2006. *WordNet: A lexical database for the english language.*

Ivan Sag, Tom Wasow, and Emily Bender. 2003. *Syntactic Theory: A Formal Introduction*. CSLI Publications, Stanford University.

James Shaw and Vasileios Hatzivassiloglou. 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.

Kees van Deemter. 2002. Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1):37–52.

Zeno Vendler. 1968. *Adjectives and Nominalizations*. Mouton.

Benjamin Lee Whorf. 1945. Grammatical categories. *Language*, 21(1):1–11.

Paul Ziff. 1960. *Semantic Analysis*. Cornell University Press, Ithaca, New York.