Referring-Expression Generation Using a Transformation-Based Learning Approach

Jill Nickerson, Stuart Shieber, and Barbara Grosz
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
{nickerso, shieber, grosz}@eecs.harvard.edu

Abstract

A natural language generation system must generate expressions that allow a reader to identify the entities to which they refer. This paper describes the creation of referring-expression (RE) generation models developed using a transformation-based learning approach. We present an evaluation of the learned models and compare their performance to the performance of a baseline system, which always generates full noun phrase REs. When compared to the baseline system, the learned models produce REs that lead to more coherent natural language documents and are more accurate and closer in length to those that people use.

Introduction

Documents generated by a computer-based natural language system must contain expressions that allow a reader to identify the entities to which they refer. We present induced models for generating referring expressions (REs) based on people's decisions. REs generated by the models can be used to create cohesive, natural-sounding documents.

Most prior work in the area of RE generation uses an incremental algorithm (Dale & Reiter 1995) to build expressions for performing initial reference. To construct an expression to refer to a given entity, the incremental algorithm searches for the set of attributes that distinguishes the given entity from all other entities with which it might be confused. For instance, a small black dog might be characterized by the two attribute-value pairs: (size:small) and (color:black). The values of these attributes are then used to build an RE for the first use of an entity.

The incremental approach has two problems: (i) it assumes that entities can be characterized in terms of a collection of attributes and their values; and (ii) it only accounts for the generation of REs that refer initially. Although the attribute-value assumption applies to some domains, in many domains, such as our test domain of car repair procedures (General Motors 2003), it is not possible to characterize entities, such as *automatic transmission shift control*, in terms of attribute-value pairs. The initial-use assumption has two limitations. First, it does not account for generating REs for repeated mentions of an entity. Second, it does not address the need to distinguish an entity from other entities of the same type. The incremental algorithm generates expressions to identify whole objects. Though this type of reference is important, there is also the need to generate expressions to refer to objects that are related in a different way: parts or components of a whole object mentioned within the context of the object. For example, in a context in which an air inlet grille panel has already been mentioned, the RE *push-in retainers* might be appropriate for referring to *air inlet grille panel push-in retainers*, a component of the air inlet grille panel.

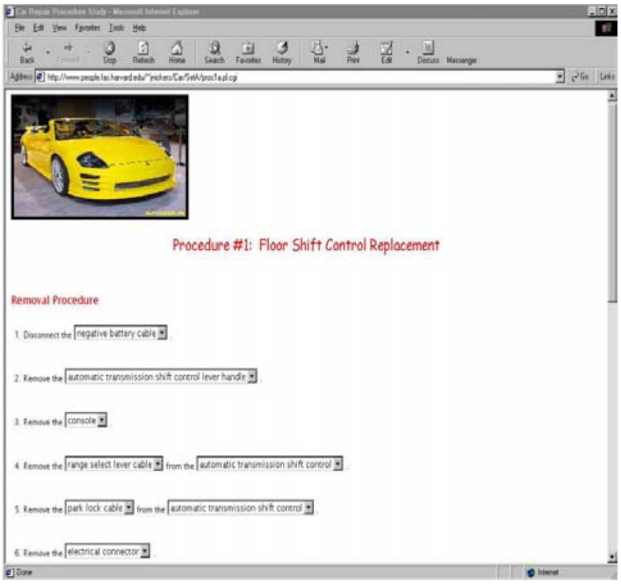
The necessity that documents not contain repetitive REs is supported by psycholinguistic studies (Gordon, Grosz, & Gilliom 1993; Gordon & Hendrick 1998, inter alia). Repeating the same RE for subsequent mentions leads to additional mental processing and, therefore, an increase in reading time. Reduced noun phrases provide a cue to coherence and lead to faster comprehension.

We present a data-driven, learning approach to the problem of RE generation. This approach overcomes the two main drawbacks of previous approaches: it does not require a hand-tailored knowledge base and it produces non-repetitive REs for both initial and repeated references. We begin by introducing a transformation-based learning algorithm. We then describe the procedure used to learn models of RE generation. The remaining sections present a performance analysis of the models. The results show that, when compared to a baseline system that always generates full noun phrases, the learned models generate REs that are more accurate and closer in length and content to those that people use.

Transformation-Based Learning

Transformation-based learning (Brill 1993) has been applied to many natural-language processing problems, including part-of-speech tagging (Brill 1995) and text chunking (Ramshaw & Marcus 1995). Input to the algorithm includes a training corpus of instances, the correct class for each instance, a baseline system, and a set of rule templates for creating possible rules. Learning a model includes the following steps:

1. Apply a baseline system to each instance in the training corpus to obtain the current corpus.
2. Use the current corpus, along with the correct class of



http://www.people.fas.harvard.edu/~pnickers/Car/SetA/proc1a.pl.cgi

Procedure #1: Floor Shift Control Replacement

Removal Procedure

1. Disconnect the [negative battery cable ▾].
2. Remove the [automatic transmission shift control lever handle ▾].
3. Remove the [console ▾].
4. Remove the [range select lever cable ▾] from the [automatic transmission shift control ▾].
5. Remove the [park lock cable ▾] from the [automatic transmission shift control ▾].
6. Remove the [electrical connector ▾].

Figure 1: Partial repair procedure used in online data-collection exercise.

the instances and the rule templates, to derive and score candidate rules. Rule templates include the combinations of features that can be used to form rules. They consist of two parts: a triggering context and a new class. To form a rule, the features are instantiated with values.

3. Select the rule with the highest score by subtracting the number of instances in the training corpus whose class the rule changes to the incorrect value from the number of instances whose class the rule changes to the correct value.
4. Append the rule to the learned rule sequence and apply the rule to obtain the current corpus.
5. Repeat Steps 2-4 until error reduction is not possible.
6. The current set of rules is the final learned sequence.

To apply the model to new data, the baseline system is used to determine the initial prediction for the class of each instance. Each rule from the learned sequence is then applied, in turn. After applying all of the rules, the class of each instance represents the learned model's prediction.

Data-Collection Exercise to Create a Corpus

To create a corpus for use in inducing models for generating REs, we conducted an online data-collection exercise (Nickerson 2005), the goal of which was to collect REs that people use. Thirty-five car repair procedures were chosen randomly from a corpus of car repair procedures.[1] A partial example of a procedure used in the exercise is shown in Figure 1.

Category	*Feature: Meaning*
Identifier	WORD: The word itself
Subsequence	NP_DIST_SAME_SUBSEQ*: #NPs since last use of NP with current prefix
	BINARY_SAME_SUBSEQ_SUBPROC: Has NP with current prefix already been used in current subprocedure?
	BINARY_SAME_SUBSEQ_SAME_STEP: Has NP with current prefix already been used in current instruction step?
Type	NP_DIST_SAME_TYPE*: #NPs since last use of NP with same head N but different word as NP current word is part of
Coreference	NP_DIST_COREF*: #NPs since last use of NP current word is part of
	STEP_DIST_COREF*: #Instruction steps since last use of NP current word is part of
	BINARY_COREF_SUBPROC: Has NP that current word is part of already been used in current subprocedure?
Syntax	BINARY_HEAD: Is the current word the head noun?
Class	REDUCE_i: Current class of word i words to right (resp., left for negative i) in NP that current word is part of
Miscellaneous	TITLE: Is current word present in title of current car repair procedure?
	COMPOUND: Is current word part of a compound NP?
	TF_IDF*: TF-IDF weight (Salton & Buckley 1988) of current prefix

Table 1: Examples of features used in rule templates.

Noun phrase REs were replaced by a pull-down menu containing two options: the full noun phrase and *other*. Subjects were encouraged to shorten REs as much as possible without introducing ambiguity. They were told to assume that the content of the procedure was correct. Choosing the first option in the menu indicated that the full noun phrase was appropriate. If subjects chose *other* from the menu, they were prompted to type in an alternate, shortened form of the full noun phrase. Thirty-five subjects (subjects were college graduates aged 25-30 who had not studied computational linguistics) participated in the study. The exercise resulted in the collection of five judgments for each of the 500 REs contained in the procedures.

RE Generation in the Paradigm of Transformation-Based Learning

Each of the 1760 words contained in the 500 REs from the data-collection exercise was taken to be an instance in the corpus. To arrive at the gold-standard class for each instance, we determined the REs that the five subjects who made judgments on it chose most frequently. This computation revealed two interesting properties. First, pronouns were used very rarely, and they were never the most frequently used expression. Second, all of the most frequently chosen REs consisted of a contiguous subsequence of words

[1] The car repair procedures in the corpus, such as the one shown in Figure 1, contained redundant unreduced REs. Therefore, the REs contained in them could not be used to induce models for generating natural-sounding REs.

```
for each $num\_bins \in \{0, 2, 3\}$
  for each $feature\_set \in \{\text{WORD}, \text{NO\_WORD}\}$
    for each $train\_num \in [1..5]$
      $learned\_rules :=$
          $learn(train\_num, num\_bins, feature\_set)$
      for each $prune \in \{ .001, .0025, .005, .01, .02,$
                             $.025, .05, .1, .15, .2, .25\}$
        $pruned\_rules :=$
            $prune\_rules(prune, learned\_rules)$
        $validation\_num := train\_num$
        $test(validation\_num, pruned\_rules)$
```

Figure 2: Learning and validating models for RE generation.

from the full noun phrase, and the last word of the subsequence was always the last noun of the full noun phrase, i.e., the head noun.

Each instance in the corpus was assigned one of two possible classes: O (outside) or I (inside).[2] The class O indicates that the RE that the subjects chose most frequently did not contain this word. All words included in the RE that subjects chose most frequently was assigned the class I. The baseline system assigns each instance the class I, indicating that the initial guess of the system is to use the full noun phrase to refer to entities.

Example features used in the rule templates are shown in Table 1. The 35 features, whose values were extracted automatically, are divided into seven categories: identifier, subsequence, type, coreference, syntax, class, and miscellaneous. Subsequence, coreference, and type features encode recency since last mention. Subsequence features are used to determine whether an object has already been referred to using the same prefix as the current one. A prefix is defined as the contiguous set of words beginning with the first word in the noun phrase (excluding determiners) and ending with the current word. Type features are used to determine the recency of different objects of the same type. Coreference features encode whether the object referred to by the noun phrase that the current word is a part of has been mentioned previously. Subsequence, type, and coreference features are motivated by prior RE generation work that keeps track of when entities in the discourse were last mentioned and whether or not an entity needs to be distinguished from other entities of the same type to generate an appropriate RE (McCoy & Strube 1999; Reiter & Dale 2000, inter alia).

The triggering context of each rule template is made up of at most three features.[3] In creating each rule, the algorithm considered 6,000-7,000 rule templates.

Models for RE Generation

Figure 2 presents the algorithm used to learn and validate models for RE generation using transformation-based learning (Nickerson 2005).[4] The algorithm uses 75% of the corpus for training and validation. The remaining 25% is used to test the final model. The training and validation set is further divided into five equal parts for 5-fold cross-validation (Weiss & Kulikowski 1991).

The value of three parameters were adjusted to create 72 parameter settings: $feature\_set$, num_bins, and $prune$. The $feature\_set$ parameter has two possible values. A value of WORD indicates that the WORD identifier feature is included in the rule templates, and NO_WORD indicates that the identifier feature is **not** used. num_bins is the number of bins used for discretizing recency features. Discretized recency features are followed by an asterisk in Table 1.[5] The $prune$ parameter[6] is used to identify generalizable rules, and not those that capture peculiarities present in the training data. The value of $prune$ represents the *p* value from a one-tailed *t* test to assess if the underlying proportion of class values that a rule correctly predicts is greater than 50% of the total number of instances to which the rule applies. The function $prune\_rules$ performs the statistical test[7] and returns the ordered set of rules for which the *p* value resulting from the statistical test is less than $prune$. For each num_bins and $feature\_set$, the algorithm calls $learn$ to learn a rule sequence. For each value of $prune$, $prune\_rules$ prunes the rule sequence. Finally, $test$ applies each pruned rule sequence. This process is performed five times, each time using a different held-out fold.

Results on the Validation Sets

To calculate performance metrics for the parameter settings, we compared the REs generated by the learned model to those chosen by the subjects in the data-collection exercise and computed average accuracy and root mean square (RMS) error over the five validation sets. Accuracy is measured by comparing the noun phrases the models generate to those most frequently chosen by subjects. An RE generated by the learned models is counted as correct only if it exactly matches the gold-standard RE. RMS error measures the deviation in length (in number of words) of the RE generated by the model from the gold-standard RE.[8] For instance, an RMS error of 1.5 indicates that, on average, REs generated by the model are within 1.5 words in length, either longer or shorter, of the gold-standard REs. The average accuracy of

[2] The method we use to assign a class to each instance is similar to the one used to apply transformation-based learning to text chunking (Ramshaw & Marcus 1995).

[3] More than three features proved to be too expensive computationally. Methods have been suggested for optimizing the rule consideration process to make it more tractable (Ramshaw & Marcus 1995).

[4] We used an existing transformation-based learning toolkit (Ngai & Florian 2001).

[5] A num_bins value of 0 indicates that discretization was not performed.

[6] We use $prune = NA$ to indicate that no pruning is performed.

[7] Prior work in the area of machine learning has used statistical tests to sort rules based on their accuracy and reliability (Yarowsky 1994; Clark & Niblett 1989, inter alia).

[8] Before calculating RMS error for the learned model, it was verified that all REs contained a contiguous sequence of nouns from the full noun phrase, with the head noun from the full noun phrase being the right-most noun in the predicted RE.

Rule	*Triggering Context*	*New Class*
1	NP_DIST_SAME_SUBSEQ $= 1 \wedge$ BINARY_SAME_SUBSEQ_SUBPROC $= 1 \wedge$ REDUCE_2 $= I$	O
2	STEP_DIST_COREF $= 1 \wedge$ BINARY_COREF_SUBPROC $= 1 \wedge$ BINARY_HEAD $= 0$	O
3	TF_IDF $= 5 \wedge$ TITLE $= 1 \wedge$ REDUCE_1 $= I$	I

Table 2: A learned rule sequence for $num\_bins = 2$, $feature\_set$ = NO_WORD, and $prune = .0025$.

Air Inlet Grill Panel Replacement
Removal Procedure

1. Open the hood.
2. Remove the wiper arm assemblies.
3. Disconnect the washer tubing from the air inlet screen.
4. Remove the air inlet grille panel push-in retainers from the air inlet grille panel.
5. Remove *the air inlet grille panel* from the vehicle.

Installation Procedure

1. Position the air inlet grille panel to the vehicle.
2. Install the air inlet grille panel push-in retainers to the air inlet grille panel.
3. Connect the washer tubing to the air inlet screen.
4. Install the wiper arm assemblies.
5. Close the hood.

Figure 3: Car repair procedure used to demonstrate application of learned rule sequence in Table 2.

Words in Noun Phrase	*Baseline Class*	*After Rule 1*	*After Rule 2*	*After Rule 3*
air	I	O	O	O
inlet	I	O	O	O
grille	I	I	O	O
panel	I	I	I	I

Table 3: Class assignment before and after application of learned rule sequence in Table 2.

the baseline model on the five validation sets is 61.09% ± 9.69%, and the average RMS error is 1.56 ± 0.49.

The performance of the learned rule sequences yielded three interesting observations. (i) In general, for $prune >$ .01, performance degrades. These rule sequences overfit the training data, and, therefore, do not generalize well to the unseen instances in the validation sets. (ii) For rule sequences in which overfitting is not a factor, models developed using $num\_bins = 2$ perform best. (iii) The WORD feature does not contribute to improving performance. Rule sequences that make use of this feature are not as general and do not perform as well as those that do not.

Table 2 presents one of the learned rule sequences for the following parameter setting: $num\_bins = 2$, $feature\_set$ = NO_WORD, and $prune = .0025$. Table 1 may be used to determine the meaning of the rules. If the current car repair procedure is the air inlet grille panel replacement procedure shown in Figure 3, and the learned sequence in Table 2 is used to generate an RE for the *air inlet grille panel* object referred to in Step 5 of the removal procedure, the baseline system is first used to assign an initial class value to each word in the noun phrase. This initial class assignment is shown in Table 3. The initial RE the baseline system generates is the full noun phrase, *air inlet grille panel*. The first and second rules in the learned sequence demonstrate the need to drop words when objects are being referred to in the locality of similar objects. Since the prefixes *air* and *air inlet* are used in the previous noun phrase (*air inlet grille panel* in Step 4), which is contained in the current subprocedure, and because the current class of *grille* and *panel* is I, the application of Rule 1 changes the class of *air* and *inlet* as shown in the table. The current RE after the application of Rule 1 is *grille panel*. The application of Rules 2 and 3 proceeds in a similar manner. The second rule applies to *air*, *inlet*, and *grille*. It does not apply to *panel*, since *panel* is the head noun of the noun phrase. The class of *grille* changes to O. The current RE is now *panel*. Finally, the third rule does not apply to any of the words in the noun phrase. The discretized TF-IDF weights of the prefixes are as follows: air:3, air inlet:2, air inlet grille:3, and air inlet grille panel:5.[9] *Panel* satisfies the first two conjuncts of the rule but fails to satisfy the last rule conjunct. The final RE generated by the rule sequence is “the *panel*”.

Developing the Final Learned Model

To determine the parameter setting used to develop the final learned model, we considered optimizing either the RMS error rate or accuracy. Figure 4 shows that RMS error rate is the metric that people optimize when making RE judgments. This figure demonstrates that, when the decision of the subjects differs from the gold-standard RE, those REs that are only off by a few words are preferred. When subjects chose an RE that is different from the gold standard, for 118 out of 244 REs, the chosen expression is within one word, either longer or shorter, of the gold standard. For 78 REs, the chosen one is within two words of the gold standard.

Because our goal is to induce a model that generates REs similar to those that people use, we, too, optimized RMS error rate. The parameter setting that led to the development of rule sequences with the lowest average RMS error rate

[9] A discretized TF-IDF weight of 5 indicates that, more so than the other prefixes, *air inlet grille panel* has a high frequency in the car repair procedure in Figure 3 and a low frequency in the entire collection of car repair procedures.

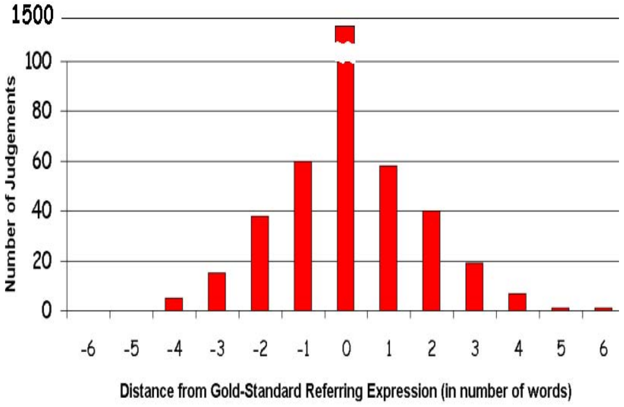


Figure 4: Distance from gold-standard RE for most frequent judgments in data-collection exercise.

is num_bins = 2, and $prune$ = .0025. Because inclusion of the identifier feature may lead to rule sequences that are not as general, we chose $feature\_set$ = NO_WORD. The average RMS error rate of the rule sequences induced using this parameter setting on the validation sets, 0.84 ± 0.15, is significantly lower than that of the baseline system, 1.56 ± 0.49 ($p <$ 0.02).

To develop the final learned model, we trained the model using 75% of the data collected from the data-collection exercise, the portion used for training and validation. To determine how well the model generalizes to unseen data, the remaining 25% of the data was used to test it. Table 4 presents the final learned rule sequence. A feature value of "$-$" indicates that the feature does not apply to the instance. The first rule indicates that if (i) a word is one of the first three words contained in a noun phrase, (ii) it is not the head of the noun phrase, and (iii) another noun phrase with the same prefix has already been mentioned in the current subprocedure, this word should be dropped. Rules 2-4 represent conditions for retaining a word in a noun phrase. Some of the conditions verified before retaining a noun include checking that the previous mention of the noun phrase is far away (Rule 2) and confirming that the noun phrase has not already been mentioned in the current procedure (Rule 4). The RMS error rate of the baseline system on the held-out test set is 1.38. For the learned rule sequence, it is 0.81 (Nickerson 2005).[10] These results indicate that the learned model, on average, generates referring expressions that are 0.57 words closer in length to those that people use.

Analysis of the Final Learned Model

Figure 5 compares the gold-standard REs with those generated by the learned model and the baseline system. The first RE in each box is the gold standard; the second, the one generated by the learned model; and the third, the baseline

Right IP Trim Plate Replacement

Removal Procedure

1. Remove	the right IP trim cover the right IP trim cover the right IP trim cover	.	
2. Remove	the right IP accessory trim plate the accessory trim plate the right IP accessory trim plate	.	
3. Remove	the right IP trim plate fasteners the plate fasteners the trim plate fasteners	from	
	the right IP trim cover area the cover area the cover area	.	
4. Gently pry	the right IP trim plate the plate the trim plate	.	

Installation Procedure

1. Snap	the right IP trim plate the right IP trim plate the right IP trim plate	into the IP.
2. Install	the right IP trim plate fasteners the plate fasteners the trim plate fasteners	to the IP.
3. Install	the right IP accessory trim plate the accessory trim plate the accessory trim plate	.
4. Install	the right IP trim cover the cover the cover	.

Figure 5: Comparison of baseline REs (first expression) and those generated by the learned model (second) with gold-standard REs (third).

system. The predictions of the model differ from the gold-standard RE for four of the nine REs contained in the repair procedure. The model predicts that, in removal procedure Step 2, the full noun phrase *right IP accessory trim plate* can be reduced to *accessory trim plate*. The gold-standard RE, however, is the full noun phrase. It is possible that more than one RE may be acceptable in a given context. In this case, for example, since the right IP accessory trim plate is being referred to in a context in which the right IP trim cover has been mentioned in the previous instruction step, reducing the full noun phrase is likely to be acceptable. Also, the subjects judged the reduction predicted by the model in this step to be licensed in a very similar context, namely Step 3 of the installation procedure. In Steps 3 and 4 of the removal procedure and Step 2 of the installation procedure, the model correctly predicts that it is possible to reduce the REs. The reduction chosen by the model in each case was chosen by subjects in the data-collection exercise, but it was not the most frequently chosen expression. Perhaps some subjects believed *trim plate* to be a compound. The fact that they reduced *right IP trim cover* to *cover* in certain contexts, however, indicates that *plate* and *plate fasteners* may be ac-

[10] The accuracy of the baseline system on the held-out test set is 59.69%; the accuracy of the learned model, 64.34%.

Rule	*Triggering Context*	*New Class*
1	BINARY_SAME_SUBSEQ_SUBPROC $= 1 \wedge$ BINARY_HEAD $= 0 \wedge$ REDUCE_3 $= -$	O
2	NP_DIST_COREF $= 2 \wedge$ REDUCE_-2 $= - \wedge$ REDUCE_1 $= I$	I
3	TF_IDF $= 2 \wedge$ REDUCE_-2 $= -$	I
4	NP_DIST_COREF $= - \wedge$ BINARY_SAME_SUBSEQ_SAME_STEP $= 0 \wedge$ COMPOUND $= 1$	I

Table 4: Final learned rule sequence for $num\_bins = 2$, $feature\_set$ $=$ NO_WORD, and $prune = .0025$.

ceptable reductions. The alternative provided by the baseline in these contexts is redundant. For example, Step 3 of the removal procedure states: Remove the *right IP trim* plate fasteners from the *right IP trim* cover area.

Once the learned model has generated REs, a post-processing step is needed to ensure that the REs unambiguously identify the correct entities. For cases in which two distinct entities of the same type have the same full form, difference words must be included in the RE generated by the model. For instance, in Figure 5, the entity of type *right IP trim cover* referred to in Step 1 of the removal procedure is the malfunctioning one, whereas the one referred to in Step 4 of the installation procedure is the new one. Difference words such as *old* and *new* must be prepended to the series of nouns predicted for inclusion in the RE generated by the model. Following post-processing, Step 4 of the removal and Step 1 of the installation procedures would read *Gently pry the* ***old*** *plate* and *Snap the* ***new*** *right IP trim plate into the IP*, respectively.

Conclusion

This paper described induced models for RE generation created using a transformation-based learning algorithm. We presented an evaluation of the models and compared their performance to a baseline system that always generates full noun phrase REs. We analyzed the results along two dimensions: length and accuracy. Our results showed that when compared to the baseline system, the REs generated by the models are more accurate and closer in length to those that people use. Further, they can be used to generate coherent, natural-sounding documents.

Acknowledgments

We thank the people at General Motors, specifically Ramasamy Uthurusamy, for providing us access to their car repair procedure corpus. The work described in this paper has been supported by the NSF under Grant Nos. IIS-9978343, IIS-9811129, and IIS-9618848. Support for the first author was also provided by an NSF Graduate Student Fellowship and a Lucent GRPW Grant.

References

Brill, E. 1993. *A corpus-based approach to language learning*. Ph.D. Dissertation, University of Pennsylvania.

Brill, E. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics* 21(4):543–566.

Clark, P., and Niblett, T. 1989. The CN2 induction algorithm. *Machine Learning* 3(4):261–283.

Dale, R., and Reiter, E. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science* 19(2):233–263.

General Motors. 2003. General Motors Service Information Web Pages. `http://service.gm.com`.

Gordon, P., and Hendrick, R. 1998. The representation and processing of coreference in discourse. *Cognitive Science* 22(4):389–424.

Gordon, P.; Grosz, B.; and Gilliom, L. 1993. Pronoun, names, and the centering of attention in discourse. *Cognitive Science* 17(3):311–347.

McCoy, K., and Strube, M. 1999. Generating anaphoric expressions: Pronoun or definite description? In *Proceedings of the Workshop on the Relation of Discourse/Dialogue Structure and Reference held in conjunction with ACL*.

Ngai, G., and Florian, R. 2001. Transformation-based learning in the fast lane. In *Proceedings of NAACL*.

Nickerson, J. 2005. *Reference specification in multilingual document production*. Ph.D. Dissertation, Harvard University. Harvard CS Technical Report TR-21-05. `http://eecs.harvard.edu/~nickerso/thesis.pdf`.

Ramshaw, L., and Marcus, M. 1995. Text chunking using transformation-based learning. In *Proceedings of ACL Workshop on Very Large Corpora*.

Reiter, E., and Dale, R. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.

Salton, G., and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management* 24(5):513–523.

Weiss, S., and Kulikowski, C. 1991. *Computer System That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann Publishers, Inc.

Yarowsky, D. 1994. A comparison of corpus-based techniques for restoring accents in Spanish and French text. In *Proceedings of ACL Workshop on Very Large Corpora*.