Automatic Evaluation of Referring Expression Generation Using Corpora *

Surabhi Gupta and Amanda J. Stent
Computer Science Department
Stony Brook University
Stony Brook, NY 11794-4400 USA
sugupta@ic.sunysb.edu, stent@cs.sunysb.edu

Abstract

We report on a set of experiments using corpora to evaluate referring expression generation for spoken dialog. In dialog, participants frequently converge on the same referring expressions even if those referring expressions are inefficient. Existing rule-based algorithms for referring expression generation do not adequately model this adaptation. We extended two such algorithms with simple models of partner adaptation. We evaluated these algorithms automatically using corpora of spoken dialog.

Referring expression generation is an important issue because referring expressions are prevalent in all types of discourse, and mproper construction of a referring expression can result in referring expressions that are ambiguous (e.g. *the book* when there are two books) or that lead to false implicatures (e.g. *the bright red book* when there is only one book) [Grice, 1975]. The generation of referring expressions is uniquely dependent on pragmatic constraints [Jordan and Walker, 2005; Krahmer and Theune, 2000]. In particular, in dialog participants adapt to each other's choice of referring expressions [Brennan, 1996; Metzing and Brennan, 2003]. However, most existing algorithms for referring expression generation do not take into account partner-specific adaptation (c.f. [Jordan and Walker, 2005]).

In this paper, we report on a set of experiments using corpora to automatically evaluate algorithms for referring expression generation for dialog. We used the MapTask and Coconut corpora, which are described in Section 2. The algorithms we use are described in Section 3. We describe our evaluation in Section 4 and conclude with some ideas for future work in Section 5.

*We would like to thank the blind reviewers for their helpful comments. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010; and by the National Science Foundation under Grant No. 0325188.

1 Related Work

Referring expression generation is one of the most widely studied research problems in natural language generation. Dale and Reiter's Incremental Algorithm [Dale and Reiter, 1995] is the most widely used. There have been several extensions to the basic Incremental Algorithm, including extensions to increase the efficiency of the referring expressions generated [Gardent and others, 2003; Horacek, 2003] and to refine the contrast set based on consideration of the discourse context [Krahmer and Theune, 2000]. Other work on referring expression generation has expanded the types of expressions that can be generated, but usually at the cost of efficiency in generation (e.g. [Gardent and others, 2003; van Deemter, 2000; 2002; Varges and van Deemter, 2005]). For these approaches, no quantitative evaluation is reported.

Siddharthan and Copestake addressed the problem of open-domain noun phrase generation [Siddharthan and Copestake, 2004]. They evaluated their algorithm by computing how many generated noun phrases were identical to corresponding human-produced noun phrases from the Wall Street Journal, and reported a performance of 81.5%.

Other work has looked at statistical noun phrase generation from corpora Poesio et al. performed several experiments on learning to predict aspects of the realization of a noun phrase. They compared the answer provided by their learned models to annotated testing data. They report an accuracy of 70% for predicting noun phrase type (e.g. personal pronoun, bare NP), and of 67.5% for predicting the syntactic form of an attribute realization given semantic and pragmatic information [Cheng *et al.*, 2001; Poesio and others, 1999]. Shaw and Hatzivassiloglu [Shaw and Hatzivassiloglou, 1999] and Malouf [Malouf, 2000] trained several models for prediction of prenominal adjective ordering; performance varies, with best performance by corpus ranging from 71.04% on financial data to 94.9% on medical data. Roy used humans to evaluate output from his shape description generator in a forced-choice experiment and reported a performance of 81.3% [Roy, 2002].

Jordan and Walker trained a classifier on an annotated version of the Coconut corpus to select sets of attributes for inclusion in generated noun phrases based on pragmatic features and information from a knowledge base [Jordan and Walker, 2005]. Their goal was to examine which of three models of noun phrase generation (include Dale and Reiter's model) works best. Their best reported classification accu-

racy (combining all models) is 59.9%. They do not perform attribute ordering.

2 Data

For our comparison of approaches to referring expression generation we used two dialog corpora, the MapTask corpus [Anderson and others, 1991] and the Coconut corpus [Di Eugenio and others, 1998]. Table 1 shows a breakdown of referring expressions in our dialogs by type. We only used non-embedded definite and indefinite noun phrases in our experiments. Furthermore, we only used noun phrases that refer to some object represented in the world of the task. For example, for MapTask references to the paper on which the map is printed such as *the left hand side* are excluded from this study.

2.1 Corpora

Maptask In each MapTask dialog [Anderson and others, 1991], there is a giver and a follower. Each participant has a map with landmarks indicated; there are differences in the position and number of landmarks on the two maps. The giver's map has a route between the landmarks; the giver provides instructions to the follower so that the follower is able to reproduce the route on the giver's map.

Due to the labor involved in annotating the dialogs, we randomly selected 30 dialogs from the MapTask corpus to use for these experiments. We used 16 dialogs where the participants had eye contact and 14 where they did not.

Coconut In each Coconut dialog [Di Eugenio and others, 1998], each partner is given a partial inventory of furniture. Their goal is to buy as much furniture as possible for two rooms of the house, the living room and the dining room. They have to try and match colors of furniture within a room. The participants are given information about calculating points for the furniture that they have bought (e.g. a sofa is worth 200 points, dining table chairs are worth 50 points each). The dialog partners sat in different rooms and communicated with each other via text messaging.

What makes this corpus interesting is that there are multiple items of furniture with the same `type` attribute, which means more scope for modifier selection. We used all 16 unique dialogs in the Coconut corpus.

2.2 Dialog processing

For each selected dialog, we constructed an annotation file, lexicon and set of knowledge representations.

Annotation Files We processed each dialog by: extracting a plain-text transcript; part-of-speech tagging it using the Brill tagger [Brill, 1992], and automatically extracting noun phrases using a pattern matcher over the part-of-speech tag sequences. We then corrected the noun phrase extraction by hand, and annotated each extracted noun phrase for referent (in the knowledge representation), type (noun phrase or pronoun), and to indicate whether the noun phrase was embedded in another noun phrase. The result was a file containing all referring expressions for the dialog in order.

Knowledge Representations We constructed a knowledge representation for each speaker in each dialog by hand from examination of the materials provided to the speakers (maps, furniture inventories). The knowledge representations for MapTask included values for six attributes for each potential referent: contents (e.g. *sandstone* in *sandstone cliff*), size (e.g. *large*), state (e.g. *parched, disused*), location, color and type (e.g. *cliffs*). For Coconut, the knowledge representations included values for five attributes for each potential referent: quantity (e.g. *2* in *2 chairs*), cost (e.g. *300* in *the $300 chair*), state (e.g. *high* in *high table*), color and type (e.g. *chair*).

Lexicons We also constructed a lexicon by hand for each dialog. The lexicon contained a list of words that might appear in a noun phrase in that dialog. We labeled each word with its word sense number from WordNet, the category of the word in our knowledge representation (e.g. size, state, location) and the singular form of the word.

2.3 Choice of corpus

The choice of corpus is particularly important for evaluation of referring expression generation. First, the corpus should be rich in the referring expressions of interest. For example, Siddharthan and Copestake found only 146 definite descriptions in the Wall Street Journal that could be used in their evaluation [Siddharthan and Copestake, 2004]. Between our two corpora, there were significant differences in terms of NP type and number and location of modifiers, as shown in Table 1. In fact, we sacrificed a focus on spoken dialog in order to obtain richer noun phrases.

Second, it is important to be clear about which noun phrases are being studied and what proportion of the number of noun phrases in the corpus they represent, so that performance improvements for competing algorithms can be evaluated in terms of global impact. We used a modified version of the disclosure table described in [Byron, 2001].

Both of our corpora involve fairly straightforward referential tasks. By contrast, in corpora of descriptions collected by psycholinguists (e.g. [Brennan, 1996; Metzing and Brennan, 2003], the conversational partners make considerable use of their general world knowledge to assign new labels to objects. These data cannot be modeled by any existing algorithm for generating referring expressions.

3 Implementation

3.1 Algorithms

We implemented Dale and Reiter's Incremental Algorithm [Dale and Reiter, 1995], and Siddharthan and Copestake's recent extension to this algorithm [Siddharthan and Copestake, 2004]. We then implemented two approaches to modeling partner effects and three approaches to pre/postmodifier placement as extensions to each algorithm. We compare the performance of all of these algorithms to each other and to a simple baseline.

We performed our experiments using a program that works through the annotation file for each dialog, passing the referent in each line to the selected algorithm and comparing the output to the human-produced referring expression. Each algorithm takes as input a referent for which to generate a referring expression, a preference list of attributes to use in generating the referring expression and a contrast set of distractors for the referent constructed from the discourse context and

	NP category	MapTask	Coconut
Total Count			
	Def	2118	116
	Indef	1411	967
	1st person pro.	563	440
	2nd person pro.	1275	165
	3rd person pro.	614	79
Total NPs		5981	1767
Excluded			
	Quantity Nouns	160	291
	Pron (first+second+third)	2452	684
	NPs not in KR, Other	2075	321
Included			
	No Modifiers	113	13
	Simple NPs	1268	229
	Complex NPs	26	242
NPs used		1294	471
Mean # attributes		2.02	3.07

Table 1: Comparison of Noun phrases in corpora used

from the knowledge representation for the current speaker. The algorithms all output an ordered list of attribute values that approximate the final referring expression.

The referent is taken from the annotation file for each dialog. The preference list is an ordered list of the attributes in the knowledge representation for that dialog. The ordering of the preference list varies by algorithm.

We look up each referent in our model of discourse context. If it is there, then the initial contrast set is the entities in the discourse context. If it is not, then the initial contrast set is the entities in the discourse context plus the entities in the current speaker's knowledge representation. After generating a referring expression, we add its referent to the discourse context. This presumes that each referring expression is perfectly understood and ignores the important role played by grounding in conversation [Clark, 1996; Cahn and Brennan, 1999]. However, in this research we did not attempt to interpret all utterances in the discourse, so this is a necessary approximation.

Baseline The initial referring expression consists solely of the value of the `type` attribute. The remaining attributes are selected at random from those attributes for which the referent has a value, until all distractors from the contrast set have been eliminated. The value of each selected attribute is prepended to the referring expression. This algorithm is equivalent to the Dale and Reiter algorithm with no ordered preference list.

Dale and Reiter Dale and Reiter's Incremental Algorithm [DR] was designed to model cognitive efficiency constraints that affect how humans produce referring expressions. The key difference from the baseline algorithm is that a preference list (an ordered list of attributes prespecified by the system designer) is used to select the next attribute for inclusion in the referring expression. As in the baseline algorithm, the value for each selected attribute is prepended to the referring expression. For the MapTask corpus, we used the following preference list: <`type, size, content, state, location, color`>. For the Coconut corpus, we used this preference list: <`type, color, cost, quantity, state`>.

Siddharthan and Copestake Siddharthan and Copestake's extension to the Incremental Algorithm [SC] was designed to permit it to be used for domains for which there do not exist detailed ontologies, especially open domain noun phrase generation. The basic algorithm is identical, except for the ordering of the preference list. For each referring expression, the attributes in the preference list are reordered so as to prefer maximally distinctive adjectives (attribute values).

Partner-Specific Adaptation In order to model partner effects, we simply consider previous mentions of the current referent. In our basic approach [DRP, SCP], we reorder the preference list for each referent to reflect the ordering of attributes in the most recent mention of that referent (by either speaker), with unmentioned attributes coming at the end of the preference list.

Partner-Specific Adaptation Variant In a variant of our basic approach to modeling partner effects [DRPV, SCPV], we not only reorder the preference list to reflect those attributes used in the most recent mention of the current referent, but also *require* that attributes used in the most recent mention be used in the referring expression to be generated, even if those attributes do not eliminate any distractors.

3.2 Generating Postmodifiers

The Incremental Algorithm only selects and orders attributes that will be used to generate noun phrase premodifiers. As Table 1 shows, noun phrases containing postmodifiers accounted for approximately 51% of the indefinite and definite noun phrases in the Coconut corpus. We implemented three extensions to this algorithm that permit the generation of complex noun phrases including postmodifiers as well as premodifiers. These extensions were all evaluated in combination with each of the six algorithms described in the previous section. For each dialog in each corpus, we used the other dialogs in that corpus to compute probabilities for these models.

Random In this approach, the decision about whether to place the attribute value before or after the head noun is made at random. The ordering of premodifiers is still right to left as specified in the preference list; the ordering of postmodifiers is left to right as specified in the preference list.

Unigram In this approach, the decision about whether to place the attribute value before or after the head noun is made by considering the relative frequency of placement of this attribute in the corpus under consideration. For example, for each of the sixteen dialogs in the Coconut corpus, we compute the relative frequencies of occurrence of each attribute before and after the head nouns in in the noun phrases of the other fifteen dialogs. During generation, we decide whether to construct a premodifier or postmodifier from a particular attribute value based on whether the relative frequency of occurrence of the attribute in premodifiers was greater than or less than the relative frequency of occurrence of the attribute in postmodifiers. The ordering of premodifiers is still right to left as specified in the preference list; the ordering of postmodifiers is left to right as specified in the preference list.

Bigram This approach also relies on knowing the relative frequencies of occurrence of the attributes in the corpus. However, in this case we compute relative frequencies in context: for each attribute, we compute the probability of occurrence of that attribute given each other attribute. During generation, we look up the relative frequencies of the current attribute occurring before and after each attribute already in the referring expression, and place the attribute value at the location of highest relative frequency. Note that this means that modifiers are not necessarily ordered as specified in the preference list; the preference list determines which modifiers are used, but the corpus frequencies determine where they are placed (cf. [Shaw and Hatzivassiloglou, 1999; Malouf, 2000]).

4 Experiments

In these experiments, we were interested in approaches to modeling partner-specific adaptation, so could not evaluate solely for comprehensibility (cf. [Roy, 2002]). Our algorithms perform attribute selection and ordering, so we could not use classification accuracy as in [Jordan and Walker, 2005]. At the same time, we were not performing surface realization, so could not use string equality (cf. [Siddharthan and Copestake, 2004]).

Our assumption was that humans are the best generators of referring expressions in conversation. The performance of a program modeling partner-specific adaptation can therefore be automatically evaluated by comparison with human-produced output. Of course, this assumption sometimes fails. For example, a human-produced referring expression may not be understood by the conversational partner; or it may be non-optimal (too long, or using attribute values that are not clear to the conversational partner). However, it is not feasible to interrupt a dialog at every referring expression to query the hearer as to whether it is "optimal".

Another issue is that two referring expressions may be equally good in context; our evaluation method does not take this into account. However, it evaluates the phenomena we are interested in (attribute selection, attribute ordering and partner-specific adaptation) and can be performed automatically, even without the use of an end-to-end dialog system. It is therefore a reasonable compromise for efficient evaluation and comparison for this task.

To evaluate our algorithms, the output referring expression (consisting of a sequence of attribute values, possibly including a determiner and almost always including `type`) is compared to the human-produced referring expression from the annotation file[1]. We automatically compute:

- (C) the number of attribute values that are correct
- (I) the number of attributes that are inserted
- (D) the number of attributes that are deleted
- (M) the number of attributes that are moved

Because we used domain-specific lexicons, we never had an incorrect value for an attribute. Our score for each generated referring expression is S = C / C + I + D + M; this score ranges from 0 to 1, with higher numbers indicating referring expressions that better match the original human-produced ones. For evaluating our algorithms, we computed the mean score over all noun phrases for each corpus.

4.1 Hypotheses

We compared the six algorithms described earlier, and our baseline algorithm, in combination with our three extensions for pre/postmodifier placement. Tables 2 and 3 show the mean scores and number of items scoring 1 for each algorithm for each corpus. We used paired t-tests to compare the performance of our algorithms. Our hypotheses were as follows:

1. DR and SC will outperform baseline
2. SC will outpeform DR
3. DRP and DRPV will outperform DR
4. SCP and SCPV will outperform SC
5. For each version of the Incremental Algorithm, Bigram will outperform Unigram which will outperform Random

There were differences in our two corpora that led to slightly different results for each one. In particular:

- For MapTask our knowledge representations were incomplete (we did not have attribute values for all attributes for all referents)
- For Coconut there were more distractors on average
- For each MapTask dialog, the two participants' knowledge representations had conflicts; for Coconut, they were complementary

MapTask

Our basic results for MapTask, based on mean scores, are:

1. DR and SC perform significantly *worse* than baseline for except for Random ($p < 0.01$).

[1] A small number were excluded at this stage because the original NP was incomplete: 20 for Maptask and 7 for Coconut.

Mean	None	Random	Unigram	Bigram
Baseline	0.636 (223)	0.622 (220)	0.636 (223)	0.636 (223)
DR	0.633 (216)	0.625 (216)	0.633 (216)	0.633 (216)
DRP	0.638 (232)	0.621 (222)	0.638 (234)	0.638 (234)
DRPV	0.764 (639)	0.657 (409)	0.760 (630)	0.764 (638)
SC	0.631 (232)	0.618 (223)	0.630 (232)	0.630 (232)
SCP	0.631 (232)	0.618 (225)	0.630 (232)	0.630 (232)
SCPV	0.760 (636)	0.664 (436)	0.755 (627)	0.762 (635)

Table 2: Mean scores and number perfect for MapTask

2. SC does *not* outperform DR. Both the algorithms give similar results except for Random, where SC performs significantly *worse* than DR ($p < 0.01$).
3. DRP performs significantly better than DR except for Random ($p < 0.01$). DRPV performs significantly better than DR for all postmodifier variants ($p < 0.01$).
4. SCP does *not* perform better than SC. However, SCPV does perform significantly better than SC for all postmodifier variants ($p < 0.01$).
5. Both Bigram and Unigram outperform Random ($p < 0.01$). Bigram outperforms Unigram for DRPV and SCPV. ($p < 0.01$).

DR does not perform significantly better than baseline because after selecting the `type` attribute, there are no remaining distractors for most of the referents in this corpus. However, the algorithms that model partner effects typically do outperform those that do not because most of the noun phrases do include at least one attribute other than type; in other words, the human-produced noun phrases are verbose.

Coconut

Our basic results for Coconut are:

1. DR and SC do perform significantly better than baseline for all postmodifier variants ($p < 0.01$).
2. SC performs significantly better than DR for all postmodifier variants ($p < 0.05$).
3. DRP and DRPV both perform significantly better than DR for all postmodifier variants ($p < 0.01$).
4. SCP does *not* perform better than SC except for Random ($p < 0.05$). SCPV performs significantly better than SC for all postmodifier variants ($p < 0.01$).
5. There is no significant difference between Random and Unigram. Bigram performs better than Random for SC, SCP and SCPV ($p < 0.05$). Bigram performs significantly better than Unigram for SC, DRP, SCP, SCPV and DRPV ($p < 0.05$).

Because there are more distractors on average for a referent in the Coconut corpus, it is less likely that simply selecting the type attribute will remove all the distractors. So, for this corpus, DR and SC perform better than baseline. Because there are more attributes per referent, there are more ways to incorrectly place an attribute value in the NP, which explains why Random and Unigram are more similar for this corpus.

For both corpora, SCP does not perform significantly better than SC, but SCPV does. We think that with SCP, the reordering of attributes leads to "over-efficient" NP generation compared to SCPV. This does not happen with DRP, for which the preference list is not re-ordered on a per-referent basis.

Discussion

Prior work in cognitive psychology has shown that humans frequently produce "non-optimal" referring expressions, and has highlighted the need to model partner effects in interpreting and producing referring expressions [Brennan, 1996; Metzing and Brennan, 2003]. The results reported here provide confirmation that our relatively simple extensions to the Incremental Algorithm can provide effective modeling of partner effects in referring expression generation, compared to state-of-the-art statistical approaches [Jordan and Walker, 2005]. These results also show that our extension to the Incremental Algorithm to permit generation of NP postmodifiers does in fact improve the algorithm's performance across both corpora.

Finally, these results show that it is important to consider the domain when selecting an algorithm for referring expression generation. When there is only one object of a particular type, our baseline algorithm performs remarkably well. While there are some important domains with multiple distractors for each possible referent (e.g. air travel, rail travel, purchasing domains in general), there are many that do not have this characteristic (e.g. customer service, tutoring, some planning domains).

There are some types of referring expression that our algorithms could not handle. First, sometimes a conversational partner seems to have invented information about an object. For example, in the MapTask corpus on some maps there is a location called "highest viewpoint", which is drawn on the map with birds flying around it. These birds were referred to as seagulls. Second, we do not generate descriptions of sets (cf. [van Deemter, 2002]). In the Coconut corpus in particular there were many complex set descriptions such as *2 green and 2 red chairs both for $100 each*. Third, we do not generate pronominal references. Adding the capability to generate third-person pronouns would require only simple modifications to our existing algorithms to track discourse focus. We know of no generation algorithm that generates first and second person pronouns; pronouns such as *we* and *they* are references to sets and so are particularly problematic.

5 Conclusions and Future Work

In this paper we analyze different approaches to generating referring expressions for spoken dialog. We show that with

Mean	None	Random	Unigram	Bigram
Baseline	0.611 (72)	0.609 (72)	0.610 (72)	0.610 (72)
DR	0.637 (74)	0.638 (74)	0.637 (74)	0.637 (74)
DRP	0.674 (112)	0.684 (115)	0.683 (115)	0.685 (115)
DRPV	0.760 (165)	0.763 (165)	0.761 (165)	0.764 (165)
SC	0.657 (114)	0.655 (114)	0.656 (114)	0.659 (114)
SCP	0.660 (121)	0.661 (121)	0.660 (121)	0.663 (121)
SCPV	0.753 (163)	0.754 (163)	0.754 (163)	0.757 (163)

Table 3: Mean scores and number perfect for Coconut

relatively simple approaches to modeling partner-specific adaptation, it is possible to improve the performance of algorithms for referring expression generation. These algorithms can also be improved by simple extensions that permit the generation of noun phrases involving postmodifiers.

We plan to integrate our implemented algorithms into a spoken dialog system for collaborative problem solving being developed partly at Stony Brook University. This will give us an opportunity to evaluate the performance of our algorithm in-use, as opposed to off-line.

References

[Anderson and others, 1991] A. Anderson et al. The HCRC Map Task corpus. *Language and Speech*, 34, 1991.

[Brennan, 1996] S. E. Brennan. Lexical entrainment in spontaneous dialog. In *Proceedings of ISSD 1996*, 1996.

[Brill, 1992] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of ANLP 1992*, 1992.

[Byron, 2001] D. Byron. The uncommon denominator. *Computational Linguistics*, 27(4), December 2001.

[Cahn and Brennan, 1999] J. Cahn and S. E. Brennan. A psychological model of grounding and repair in dialog. In *Proceedings of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, 1999.

[Cheng *et al.*, 2001] H. Cheng, M. Poesio, R. Henschel, and C. Mellish. Corpus-based NP modifier generation. In *Proceedings of NAACL 2001*, 2001.

[Clark, 1996] H. Clark. *Using Language*. Cambridge University Press, 1996.

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

[Di Eugenio and others, 1998] B. Di Eugenio et al. An empirical investigation of proposals in collaborative dialogues. In *Proceedings of COLING-ACL 1998*, 1998.

[Gardent and others, 2003] C. Gardent et al. Generating definite descriptions: Non-incrementality, inference, and data. In T. Pechman and C. Habel, editors, *Multidisciplinary approaches to language production*. Walter de Gruyter, Berlin, 2003.

[Grice, 1975] H. P. Grice. Logic and conversation. In *Syntax and Semantics: Vol 3, Speech Acts*. Academic Press., New York, 1975.

[Horacek, 2003] H. Horacek. A best-first search algorithm for generating referring expressions. In *Proceedings of ACL 2003*, 2003.

[Jordan and Walker, 2005] P. Jordan and M. Walker. Learning content selection rules for generating object descriptions in dialogue. *Journal of Artificial Intelligence Research*, 4, 2005.

[Krahmer and Theune, 2000] E. Krahmer and M. Theune. Efficient context-sensitive generation of referring expressions. In K. van Deemter and R. Kibble, editors, *Information Sharing: Givenness and Newness in Language Processing*. CSLI Publications, 2000.

[Malouf, 2000] R. Malouf. The order of prenominal adjectives in natural language generation. In *Proceedings of ACL 2000*, 2000.

[Metzing and Brennan, 2003] C. Metzing and S.E. Brennan. When conceptual pacts are broken. *Journal of Memory and Language*, 49, 2003.

[Poesio and others, 1999] M. Poesio et al. Statistical NP generation: A first report. In *Proceedings of the ESSLLI Workshop on NP Generation*, 1999.

[Roy, 2002] D. Roy. Learning visually grounded words and syntax for a scene description task. *Computer Speech and Language special issue on Spoken Language Generation*, 16(3–4), 2002.

[Shaw and Hatzivassiloglou, 1999] J. Shaw and V. Hatzivassiloglou. Ordering among premodifiers. In *Proceedings of ACL 1999*, 1999.

[Siddharthan and Copestake, 2004] A. Siddharthan and A. Copestake. Generating referring expressions in open domains. In *Proceedings of ACL 2004*, 2004.

[van Deemter, 2000] K. van Deemter. Generating vague descriptions. In *Proceedings of INLG 2000*, 2000.

[van Deemter, 2002] K. van Deemter. Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1), March 2002.

[Varges and van Deemter, 2005] S. Varges and K. van Deemter. Generating referring expressions containing quantifiers. In *Proceedings of IWCS-6*, 2005.