Trainable Speaker-Based Referring Expression Generation

Giuseppe Di Fabbrizio and **Amanda J. Stent** and **Srinivas Bangalore**
AT&T Labs - Research, Inc.
180 Park Avenue
Florham Park, NJ 07932, USA
{pino,stent,srini}@research.att.com

Abstract

Previous work in referring expression generation has explored general purpose techniques for attribute selection and surface realization. However, most of this work did not take into account: a) stylistic differences between speakers; or b) trainable surface realization approaches that combine semantic and word order information. In this paper we describe and evaluate several end-to-end referring expression generation algorithms that take into consideration speaker style and use data-driven surface realization techniques.

1 Introduction

Natural language generation (NLG) systems have typically decomposed the problem of generating a linguistic expression from a conceptual specification into three major steps: *content planning*, *text planning* and *surface realization* (Reiter and Dale, 2000). The task in content planning is to select the information that is to be conveyed to maximize communication efficiency. The task in text planning and surface realization is to use the available linguistic resources (words and syntax) to convey the selected information using well-formed linguistic expressions.

During a discourse (whether written or spoken, monolog or dialog), a number of entities are introduced into the discourse context shared by the reader/hearer and the writer/speaker. Constructing linguistic references to these entities efficiently and effectively is a problem that touches on all parts of an NLG system. Traditionally, this problem is split into two parts. The task of selecting the attributes to use in referring to an entity is the *attribute selection* task, performed during content planning or sentence planning. The actual construction of the referring expression is part of *surface realization*.

There now exist numerous general-purpose algorithms for attribute selection (e.g., (Dale and Reiter, 1995; Krahmer et al., 2003; Belz and Gatt, 2007; Siddharthan and Copestake, 2004)). However, these algorithms by-and-large focus on the algorithmic aspects of referring expression generation rather than on psycholinguistic factors that influence language production. For example, we know that humans exhibit individual differences in language production that can be quite pronounced (e.g. (Belz, 2007)). We also know that the language production process is subject to *lexical priming*, which means that words and concepts that have been used recently are likely to appear again (Levelt, 1989).

In this paper, we look at attribute selection and surface realization for referring expression generation using the TUNA corpus [1], an annotated corpus of human-produced referring expressions that describe furniture and people. We first explore the impact of individual style and priming on attribute selection for referring expression generation. To get an idea of the potential improvement when modeling these factors, we implemented a version of full brevity search that uses speaker-specific constraints, and another version that also uses recency constraints. We found that using speaker-specific constraints led to big performance gains for both TUNA domains, while the use of re-

[1] http://www.csd.abdn.ac.uk/research/tuna/

CoNLL 2008: Proceedings of the 12th Conference on Computational Natural Language Learning, pages 151–158
Manchester, August 2008

cency constraints was not as effective for TUNA-style tasks. We then modified Dale and Reiter's classic attribute selection algorithm (Dale and Reiter, 1995) to model individual differences in style, and found performance gains in this more greedy approach as well.

Then, we look at surface realization for referring expression generation. There are several approaches to surface realizations described in the literature (Reiter and Dale, 2000) ranging from hand-crafted template-based realizers to data-driven syntax-based realizers (Langkilde and Knight, 2000; Bangalore and Rambow, 2000). Template-based realization provides a straightforward method to fill out pre-defined templates with the current attribute values. Data-driven syntax-based methods employ techniques that incorporate the syntactic relations between words which can potentially go beyond local adjacency relations. Syntactic information also helps in eliminating ungrammatical sentence realizations. At the other extreme, there are techniques that exhaustively generate possible realizations with recourse to syntax in as much as it is reflected in local n-grams. Such techniques have the advantage of being robust although they are inadequate to capture long-range dependencies. We explore three techniques for the task of referring expression generation that are different hybrids of hand-crafted and data-driven methods.

The layout of this paper is as follows: In Section 2, we describe the TUNA data set and the task of identifying target entities in the context of distractors. In Section 3, we present our algorithms for attribute selection. Our algorithms for surface realization are presented in Section 4. Our evaluation of these methods for attribute selection and surface realization are presented in Sections 5 and 6.

2 The TUNA Corpus

The TUNA corpus was constructed using a web-based experiment. Participants were presented with a sequence of web pages, on each of which they saw displayed a selection of 7 pictures of either furniture (e.g. Figure 1) or people (e.g. Figure 2) sparsely placed on a 3 row x 5 column grid. One of the pictures (the *target*) was highlighted; the other 6 objects (the *distractors*) were randomly selected from the object database. Participants were told that they were interacting with a computer system to remove all but the highlighted picture from the screen. They entered a description of the object using natural language to identify the object to the computer system.

The section of the TUNA corpus we used was that provided for the REG 2008 Challenge[2]. The training data includes 319 referring expressions in the furniture domain and 274 in the people domain. The development data (which we used for testing) includes 80 referring expressions in the furniture domain and 68 in the people domain.

Figure 1: Example of data from the furniture domain (*The red couch on top*).

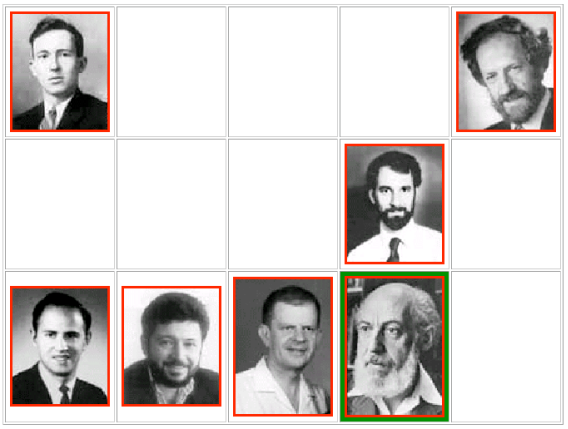

Figure 2: Example of data from the people domain (*The bald subject on the bottom with the white beard*).

3 Attribute Selection Algorithms

Given a set of entities with attributes appropriate to a domain (e.g., cost of flights, author of a book,

[2]http://www.nltg.brighton.ac.uk/research/reg08/. Preliminary versions of these algorithms were used in this challenge and presented at INLG 2008.

color of a car) that are in a discourse context, and a target entity that needs to be identified, the task of attribute selection is to select a subset of the attributes that uniquely identifies the target entity. (Note that there may be more than one such attribute set.) The efficacy of attribute selection can be measured based on the minimality of the selected attribute set as well as its ability to determine the target entity uniquely. There are variations however in terms of what makes an attribute set more preferable to a human. For example, in a people identification task, attributes of faces are generally more memorable than attributes pertaining to outfits. In this paper, we demonstrate that the attribute set is speaker dependent.

In this section, we present two different attribute selection algorithms. The **Full Brevity** algorithm selects the attribute set by exhaustively searching through all possible attribute sets. In contrast, **Dale and Reiter** algorithm orders the attributes based on a heuristic (motivated by human preference) and selects the attributes in that order until the target entity is uniquely determined. We elaborate on these algorithms below.

Full Brevity (FB) We implemented a version of full brevity search. It does the following: first, it constructs $\mathcal{AS}$, the set of attribute sets that uniquely identify the referent given the distractors. Then, it selects an attribute set $AS_u \in \mathcal{AS}$ based on one of the following four criteria: 1) The **minimality (FB-m)** criterion selects from among the smallest elements of $\mathcal{AS}$ at random. 2) The **frequency (FB-f)** criterion selects the element of $\mathcal{AS}$ that occurred most often in the training data. 3) The **speaker frequency (FB-sf)** criterion selects the element of $\mathcal{AS}$ used most often by this speaker in the training data, backing off to FB-f if necessary. This criterion models individual speaking/writing style. 4) Finally, the **speaker recency (FB-sr)** criterion selects the element of $\mathcal{AS}$ used most recently by this speaker in the training data, backing off to FB-sf if necessary. This criterion models priming.

Dale and Reiter We implemented two variants of the classic Dale & Reiter attribute selection (Dale and Reiter, 1995) algorithm. For **Dale & Reiter basic (DR-b)**, we first build the preferred list of attributes by sorting the attributes according to frequency of use in the training data. We keep separate lists based on the "LOC" condition (if its value was "+LOC", the participants were told that they could refer to the target using its location on the screen; if it was "-LOC", they were instructed not to use location on the screen) and backoff to a global preferred attribute list if necessary. Next, we iterate over the list of preferred attributes and select the next one that rules out at least one entity in the contrast set until no distractors are left. **Dale & Reiter speaker frequency (DR-sf)** uses a different preferred attribute list for each speaker, backing off to the DR-b preferred list if an attribute has never been observed in the current speaker's preferred attribute list. For the purpose of this task, we did not use any external knowledge (e.g. taxonomies).

4 Surface Realization Approaches

A surface realizer for referring expression generation transforms a set of attribute-value pairs into a linguistically well-formed expression. Our surface realizers, which are all data-driven, involve four stages of processing: (a) lexical choice of words and phrases to realize attribute values; (b) generation of a space of surface realizations (T); (c) ranking the set of realizations using a language model (LM); (d) selecting the best scoring realization. In general, the best ranking realization (T^*) is described by equation 1:

$$T^* = Bestpath(Rank(T, LM)) \qquad (1)$$

We describe three different methods for creating the search space of surface realizations – *Template-based*, *Dependency-based* and *Permutation-based* methods. Although these techniques share the same method for ranking, they differ in the methods used for generating the space of possible surface realizations.

4.1 Generating possible surface realizations

In order to transform the set of attribute-value pairs into a linguistically well-formed expression, the appropriate words that realize each attribute value need to be selected (lexical choice) and the selected words need to be ordered according to the syntax of the target language (lexical order). We present different models for approximating the syntax of the target language. All three models tightly integrate the lexical choice and lexical reordering steps.

4.1.1 Template-Based Realizer

In the template-based approach, surface realizations from our training data are used to infer a set of templates. In the TUNA data, each attribute in each referring expression is annotated with its attribute type (e.g. in "the large red sofa" the second word is labeled 'size', the third 'color' and the fourth 'type'). We extract the annotated referring expressions from each trial in the training data and replace each attribute value with its type (e.g. "the `size color type`") to create a template. Each template is indexed by the lexicographically sorted list of attribute types it contains (e.g. `color size type`). If an attribute set is not found in the training data (e.g. `color size`) but a superset of that set is (e.g. `color size type`), then the corresponding template(s) may be used, with the un-filled attribute types deleted prior to output.

At generation time, we find all possible realizations (l) (from the training data) of each attribute value (a) in the input attribute set ($\mathcal{AS}$), and fill in each possible template (t) with each combination of the attribute realizations. The space of possible surface realizations is represented as a weighted finite-state automaton. The weights are computed from the prior probability of each template and the prior probability of each lexical item realizing an attribute (Equation 2). We have two versions of this realizer: one with speaker-specific lexicons and templates (**Template-S**), and one without (**Template**). We report results for both.

$$P(T|AS) = \sum_{t} P(t|AS) * \prod_{a \in t} \sum_{l} P(l|a,t) \quad (2)$$

4.1.2 Dependency-Based Realizer

To construct our dependency-based realizer, we first parse all the word strings from the training data using the dependency parser described in (Bangalore et al., 2005; Nasr and Rambow, 2004). Then, for every pair of words w_i, w_j that occur in the same referring expression (RE) in the training data, we compute: $freq(i < j)$, the frequency with which w_i precedes w_j in any RE; $freq(dep(w_i, w_j) \wedge i < j)$, the frequency with which w_i depends on and precedes w_j in any RE, and $freq(dep(w_i, w_j) \wedge j < i)$, the frequency with which w_i depends on and follows w_j in any RE.

At generation time, we find all possible realizations of each attribute value in the input attribute set, and for each combination of attribute realizations, we find the most likely set of dependencies and precedences given the training data. In other words, we bin the selected attribute realizations according to whether they are most likely to precede, depend on and precede, depend on and follow, or follow, the head word they are closest to. The result is a set of weighted partial orderings on the attribute realizations. As with the template-based surface realizer, we implemented speaker-specific and speaker-independent versions of the dependency-based surface realizer. Once again, we encode the space of possible surface realizations as a weighted finite-state automaton.

4.1.3 Permute and Rank Realizer

In this method, the lexical items associated with each attribute value to be realized are treated as a disjunctive set of tokens. This disjunctive set is represented as a finite-state automaton with two states and transitions between them labeled with the tokens of the set. The transitions are weighted by the negative logarithm of the probability of the lexical token (l) being associated with that attribute value (a): $(-log(P(l|a)))$. These sets are treated as bags of tokens; we create permutations of these bags of tokens to represent the set of possible surface realizations.

In general, the number of states of the minimal permutation automaton of even a linear automaton (finite-state representation of a string) grows exponentially with the number of words of the string. Although creating the full permutation automaton for full natural language generation tasks could be computationally prohibitive, most attribute sets in our two domains contain no more than five attributes. So we choose to explore the full permutation space. A more general approach might constrain permutations to be within a local window of adjustable size (also see (Kanthak et al., 2005)).

Figure 3 shows the minimal permutation automaton for an input sequence of 4 words and a window size of 2. Each state of the automaton is indexed by a bit vector of size equal to the number of words/phrases of the target sentence. Each bit of the bit vector is set to 1 if the word/phrase in that bit position is used on any path from the initial to the current state. The next word for permutation from a given state is restricted to be within the window size (2 in our case) positions counting from the first as-yet uncovered position in that state. For example, the state indexed with vector "1000" rep-

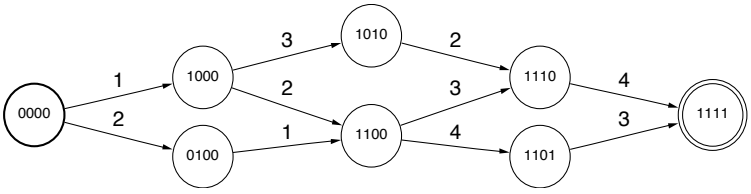

Figure 3: Locally constraint permutation automaton for a sentence with 4 positions and a window size of 2.

resents the fact that the word/phrase at position 1 has been used. The next two (window=2) positions are the possible outgoing arcs from this state with labels 2 and 3 connecting to state "1100" and "1010" respectively. The bit vectors of two states connected by an arc differ only by a single bit. Note that bit vectors elegantly solve the problem of recombining paths in the automaton as states with the same bit vectors can be merged. As a result, a fully minimized permutation automaton has only a single initial and final state.

4.2 Ranking and Recovering a Surface Realization

These three methods for surface realization create a space of possible linguistic expressions given the set of attributes to be realized. These expressions are encoded as finite-state automata and have to be ranked based on their syntactic well-formedness. We approximate the syntactic well-formedness of an expression by the n-gram likelihood score of that expression. We use a trigram model trained on the realizations in the training corpus. This language model is also represented as a weighted finite-state automaton. The automaton representing the space of possible realizations and the one representing the language model are composed. The result is an automaton that ranks the possible realizations according to their n-gram likelihood scores. We then produce the best-scoring realization as the target realization of the input attribute set.

We introduce a parameter λ which allows us to control the importance of the prior score relative to the language model scores. We weight the finite-state automata according to this parameter as shown in Equation 3.

$$T^* = Bestpath(\lambda * T \circ (1 - \lambda) * LM) \quad (3)$$

	DICE	MASI	Acc.	Uniq.	Min.
	Furniture				
FB-m	.36	.16	0	1	1
FB-f	.81	.58	.40	1	0
FB-sf	.95	.87	.79	1	0
FB-sr	.93	.81	.71	1	0
DR-b	.81	.60	.45	1	0
DR-sf	.86	.64	.45	1	.04
	People				
FB-m	.26	.12	0	1	1
FB-f	.58	.37	.28	1	0
FB-sf	.94	.88	.84	1	.01
FB-sr	.93	.85	.79	1	.01
DR-b	.70	.45	.25	1	0
DR-sf	.78	.55	.35	1	0
	Overall				
FB-m	.32	.14	0	1	1
FB-f	.70	.48	.34	1	0
FB-sf	.95	.87	.81	1	.01
FB-sr	.93	.83	.75	1	.01
DR-b	.76	.53	.36	1	0
DR-sf	.82	.60	.41	1	.02

Table 1: Results for attribute selection

5 Attribute Selection Experiments

Data Preparation The training data were used to build the models outlined above. The development data were then processed one-by-one.

Metrics We report performance using the metrics used for the REG 2008 competition. The MASI metric is a metric used in summarization that measures agreement between two annotators (or one annotator and one system) on set-valued items (Nenkova et al., 2007). Values range from 0 to 1, with 1 representing perfect agreement. The DICE metric is also a measure of association whose value varies from 0 (no association) to 1 (total association) (Dice, 1945). The Accuracy metric is binary-valued: 1 if the attribute set is identical to that selected by the human, 0 otherwise. The Uniqueness metric is also binary-valued: 1 if the attribute set uniquely identifies the target referent among the distractors, 0 otherwise. Finally, the Minimality metric is 1 if the selected attribute set is as small as possible (while still uniquely identifying the target referent), and 0 otherwise. We note

that attribute selection algorithms such as Dale & Reiter's are based on the observation that humans frequently do not produce minimal referring expressions.

Results Table 1 shows the results for variations of full brevity. As we would expect, all approaches achieve a perfect score on uniqueness. For both corpora, we see a large performance jump when we use speaker constraints for all metrics other than minimality. However, when we incorporate recency constraints as well performance declines slightly. We think this is due to two factors: first, the speakers are not in a conversation, and self-priming may have less impact than other-priming; and second, we do not always have the most recent prior utterance for a given speaker in the training data.

Table 1 also shows the results for variations of Dale & Reiter's algorithm. When we incorporate speaker constraints, we again see a performance jump for most metrics, although compared to the best possible case (full brevity) there is still room for improvement.

We conclude that speaker constraints can be successfully used in standard attribute selection algorithms to improve performance on this task.

The most relevant previous research is the work of (Gupta and Stent, 2005), who modified Dale and Reiter's algorithm to model speaker adaptation in dialog. However, this corpus does not involve dialog so there are no cross-speaker constraints, only within-speaker constraints (speaker style and priming).

6 Surface Realization Experiments

Data Preparation We first normalized the training data to correct misspellings and remove punctuation and capitalization. We then extracted a phrasal lexicon. For each attribute value we extracted the count of all realizations of that value in the training data. We treated locations as a special case, storing separately the realizations of x-y coordinate pairs and single x- or y-coordinates. We added a small number of realizations by hand to cover possible attribute values not seen in the training data.

Realization We ran two realization experiments. In the first experiment, we used the human-selected attribute sets in the development data as the input to realization. If we want to maximize humanlikeness, then using these attribute sets should give us an idea of the best possible performance of our realization methods. In the second experiment, we used the attribute sets output by our best-performing attribute selection algorithms (FB-sf and DR-sf) as the input to realization.

	λ	SED	ACC	Bleu	NIST
	Furniture				
Permute&Rank	0.01	3.54	0.14	0.311	3.87
Dependency	0.90	4.51	0.09	0.206	3.29
Dependency-S	0.60	4.30	0.11	0.232	3.91
Template	0.10	3.59	0.13	0.328	3.93
Template-S	0.10	2.80	0.28	0.403	4.67
	People				
Permute&Rank	0.04	4.37	0.10	0.227	3.15
Dependency	0.70	6.10	0.00	0.072	2.35
Dependency-S	0.50	5.84	0.02	0.136	3.05
Template	0.80	3.87	0.07	0.250	3.18
Template-S	0.70	3.79	0.15	0.265	3.59
	Overall				
Permute&Rank	.01/.04	3.92	0.12	0.271	4.02
Dependency	0.9/0.7	5.24	0.05	0.146	3.23
Dependency-S	0.6/0.5	5.01	0.07	0.187	3.98
Template	0.1/0.8	3.77	0.10	0.285	4.09
Template-S	0.1/0.7	3.26	0.22	0.335	4.77

Table 2: Results for realization using speakers' attribute selection (SED: String Edit Distance, ACC: String Accuracy)

Metrics We report performance of our surface realizers using the metrics used for the REG 2008 shared challenge and standard metrics used in the natural language generation and machine translation communities. String Edit Distance (SED) is a measure of the number of words that would have to be added, deleted, or replaced in order to transform the generated referring expression into the one produced by the human. As used in the REG 2008 shared challenge, it is unnormalized, so its values range from zero up. Accuracy (ACC) is binary-valued: 1 if the generated referring expression is identical to that produced by the human (after spelling correction and normalization), and 0 otherwise. Bleu is an n-gram based metric that counts the number of 1, 2 and 3 grams shared between the generated string and one or more (preferably more) reference strings (Papenini et al., 2001). Bleu values are normalized and range from 0 (no match) to 1 (perfect match). Finally, the NIST metric is a variation on the Bleu metric that, among other things, weights rare n-grams higher than frequently-occurring ones (Doddington, 2002). NIST values are unnormalized.

	SED		ACC		Bleu		NIST	
	Furniture							
	FB-sf	DR-sf	FB-sf	DR-sf	FB-sf	DR-sf	FB-sf	DR-sf
Permute&Rank	3.97	4.22	0.09	0.06	.291	.242	3.82	3.32
Dependency	4.80	5.03	0.04	0.03	.193	.105	3.32	2.46
Dependency-S	4.71	4.88	0.06	0.04	.201	.157	3.74	3.26
Template	3.89	4.56	0.09	0.05	.283	.213	3.48	3.22
Template-S	3.26	3.90	0.19	0.12	.362	.294	4.41	4.07
	People							
Permute&Rank	4.75	5.82	0.09	0.03	.171	.110	2.70	2.31
Dependency	6.35	6.91	0.00	0.00	.068	.073	1.81	1.86
Dependency-S	5.94	6.18	0.01	0.00	.108	.113	2.73	2.41
Template	3.62	4.24	0.07	0.04	.231	.138	2.88	1.35
Template-S	3.76	4.38	0.12	0.06	.201	.153	2.76	1.88
	Overall							
Permute&Rank	4.33	4.96	0.09	0.05	.236	.235	3.73	3.72
Dependency	5.51	6.00	0.02	0.01	.136	.091	2.97	2.50
Dependency-S	5.36	5.67	0.04	0.02	.159	.136	3.77	3.25
Template	3.76	4.41	0.08	0.05	.258	.180	3.69	2.89
Template-S	3.48	4.12	0.16	0.09	.288	.229	4.15	3.58

Table 3: Results for realization with different attribute selection algorithms

	Furniture		People	
	FB-sf	DR-sf	FB-sf	DR-sf
Permute&Rank	.01	.05	.05	.04
Dependency	.9	.9	.9	.1
Dependency-S	.2	.2	.4	.4
Template	.8	.8	.8	.8
Template-S	.6	.8	.8	.8

Table 4: Optimal λ values with different attribute selection algorithms

Results Our experimental results are shown in Tables 2 and 3. (These results are the results obtained with the language model weighting that gives best performance; the weights are shown in Tables 2 and 4.) Our approaches work better for the furniture domain, where there are fewer attributes, than for the people domain. For both domains, for automatic and human attribute selection, the speaker-dependent Template-based approach seems to perform the best, then the speaker-independent Template-based approach, and then the Permute&Rank approach. However, we find automatic metrics for evaluating generation quality to be unreliable. We looked at the output of the surface realizers for the two examples in Section 2. The best output for the example in Figure 1 is from the FB-sf template-based speaker-dependent algorithm, which is *the big red sofa*. The worst output is from the DR-sf dependency-based speaker-dependent algorithm, which is *on the left red chair with three seats*. The best output for the example in Figure 2 is from the FB-sf template-based speaker-independent algorithm, which is *the man with the white beard*. The worst output is from the FB-sf dependency-based speaker-dependent algorithm, which is *beard man white*.

Discussion The Template-S approach achieves the best string edit distance scores, but it is not very robust. If no examples are found in the training data that realize (a superset of) the input attribute set, neither Template approach will produce any output.

The biggest cause of errors for the Permute and Reorder approach is missing determiners and missing modifiers. The biggest cause of errors for the Dependency approach is missing determiners and reordered words. The Template approach sometimes has repeated words (e.g. "middle", where "middle" referred to both x- and y-coordinates).

Here we report performance using automatic metrics, but we find these metrics to be unreliable (particularly in the absence of multiple reference texts). Also, we are not sure that people would accept from a computer system output that is very human-like in this domain, as the human-like output is often ungrammatical and telegraphic (e.g. "grey frontal table"). We plan to do a human evaluation soon to better analyze our systems' performance.

7 Conclusions

When building computational models of language, knowledge about the factors that influence human language production can prove very helpful. This knowledge can be incorporated in frequentist and heuristic approaches as constraints or features. In the experiments described in this paper, we used

data-driven, speaker-aware approaches to attribute selection and referring expression realization. We showed that individual speaking style can be usefully modeled even for quite 'small' generation tasks, and confirmed that data-driven approaches to surface realization can work well using a range of lexical, syntactic and semantic information.

We plan to explore the impact of human visual search strategies (Rayner, 1998) on the referring expression generation task. In addition, we are planning a human evaluation of the generation systems' output. Finally, we plan to apply our algorithms to a conversational task.

Acknowledgments

We thank Anja Belz, Albert Gatt, and Eric Kow for organizing the REG competition and providing data, and Gregory Zelinsky for discussions about visually-based constraints.

References

Bangalore, S. and O. Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proc. COLING*.

Bangalore, S., A. Emami, and P. Haffner. 2005. Factoring global inference by enriching local representations. Technical report, AT&T Labs-Research.

Belz, A. and A. Gatt. 2007. The attribute selection for GRE challenge: Overview and evaluation results. In *Proc. UCNLG+MT at MT Summit XI*.

Belz, A. 2007. Probabilistic generation of weather forecast texts. In *Proc. NAACL/HLT*.

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

Dice, L. 1945. Measures of the amount of ecologic association between species. *Ecology*, 26.

Doddington, G. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. HLT*.

Gupta, S. and A. Stent. 2005. Automatic evaluation of referring expression generation using corpora. In *Proc. UCNLG*.

Kanthak, S., D. Vilar, E. Matusov, R. Zens, and H. Ney. 2005. Novel reordering approaches in phrase-based statistical machine translation. In *Proc. ACL Workshop on Building and Using Parallel Texts*.

Krahmer, E., S. van Erk, and A. Verleg. 2003. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1).

Langkilde, I. and K. Knight. 2000. Forest-based statistical sentence generation. In *Proc. NAACL*.

Levelt, W., 1989. *Speaking: From intention to articulation*, pages 222–226. MIT Press.

Nasr, A. and O. Rambow. 2004. Supertagging and full parsing. In *Proc. 7th International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7)*.

Nenkova, A., R. Passonneau, and K. McKeown. 2007. The Pyramid method: incorporating human content selection variation in summarization evaluation. *ACM Transactions on speech and language processing*, 4(2).

Papenini, K., S. Roukos, T. Ward, and W.-J. Zhu. 2001. BLEU: A method for automatic evaluation of machine translation. In *Proc. ACL*.

Rayner, K. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3).

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

Siddharthan, A. and A. Copestake. 2004. Generating referring expressions in open domains. In *Proc. ACL*.