Referring Expression Generation Using Speaker-based Attribute Selection and Trainable Realization (ATTR)

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

In the first REG competition, researchers proposed several general-purpose algorithms for attribute selection for referring expression generation. 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

There now exist numerous general-purpose algorithms for attribute selection used in referring expression generation (e.g., (Dale and Reiter, 1995; Krahmer et al., 2003; Belz and Gatt, 2007)). 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 style differences during 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 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 (Dale, 1992) 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 recency 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 speaker-specific constraints, and found performance gains in this more greedy approach as well.

Then we looked at surface realization for referring expression generation. There are several approaches to surface realization 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 involves the insertion of attribute values into predetermined templates. Data-driven syntax-based methods use syntactic relations between words (including long-distance relations) for word ordering. Other data-driven techniques 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. In this paper, we explore three techniques for the task of referring expression generation that are different hybrids of hand-crafted and data-driven methods.

The remainder of this paper is organized as follows: In Section 2, we present the algorithms for attribute selection. The different methods for surface realizers are presented in Section 3. The experiments concerning the attribute selection and surface realization are presented in Section 4 and Section 5. The final remarks are discussed in Section 6.

2 Attribute Selection Algorithms

Full Brevity (FB) We implemented a version of full brevity search (Dale, 1992). 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 a selection criterion. The **minimality (FB-m)** criterion selects from among the smallest elements of $\mathcal{AS}$ at random. The **frequency (FB-f)** criterion selects from among the elements of $\mathcal{AS}$ the one that occurred most often in the training data. The **speaker frequency (FB-sf)** criterion selects from among the elements of $\mathcal{AS}$ the one used most often by this speaker in the training data, backing off to FB-f if necessary. This criterion models speaker-specific constraints. Finally, the **speaker recency (FB-sr)** criterion selects from among the elements of $\mathcal{AS}$ the one used most recently by this speaker in the training data, backing off to FB-sf if necessary. This criterion models priming and speaker-specific constraints.

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 most frequently used attributes in the training set. We keep separate lists based upon the "+LOC" and "-LOC" conditions and backoff to a global preferred frequency list in case the attributes are not covered in the current list (merge and sort by frequency). 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. The **Dale & Reiter speaker frequency (DR-sf)** uses a speaker-specific preferred list, backing off to the DR-b preferred list if an attribute is not in the current speaker's preferred list. For this task, we ignored any further attribute knowledge base or taxonomy abstraction.

3 Surface Realization Approaches

We summarize our approaches to surface realization in this section. All three surface realizers have the same four stages: (a) lexical choice of words and phrases for the 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.

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

Template-Based Realizer To construct our template-based realizer, we extract the annotated word string from each trial in the training data and replace each annotated text segment with the attribute type with which it is annotated. The key for each template is the lexicographically sorted list of attribute types it contains. Consequently, any attribute lists not found in the training data cannot be realized by the template-based realizer; however, if there is a template for an input attribute list it is quite likely to be coherent.

At generation time, we find all possible realizations of each attribute in the input attribute set, and fill in each possible template with each combination of the attribute realizations. We report results for two versions of this realizer: one with speaker-specific lexicon and templates (**Template-S**), and one without (**Template**).

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; $freq(i = j - 1)$, the frequency with which w_i immediately 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 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.

Permute and Rank In this method, the lexical items associated with each of the 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 (w) being associated with that attribute value ($attr$): ($-log(P(w|attr))$). These sets are treated as unordered bags of tokens; we create permutations of these bags of tokens to represent the set of possible surface realizations. We then use the language model to rank this set of possible realizations and recover the highest scoring RE.

	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

Unfortunately, the number of states of the minimal permutation automaton of even a linear automata (finite-state machine representation of a string) grows exponentially with the number of words of the string. So, instead of creating a full permutation automaton, we choose to constrain permutations to be within a local window of adjustable size (also see (Kanthak et al., 2005)).

4 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. For our final submissions, we use training and development data to build our models.

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. 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; 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 and Reiter's algorithm. When we incorporate speaker constraints, we again see a performance jump, although compared to the best possible case (full brevity) there is still room for improvement.

	String-Edit Dist.			Accuracy		
	Furniture					
	DEV	FB-sf	DR-sf	DEV	FB-sf	DR-sf
Permute&Rank	4.39	4.60	4.74	0.07	0.04	0.03
Dependency	3.90	4.25	5.50	0.14	0.06	0.03
Template	4.36	4.33	5.39	0.07	0.05	0.03
Template-S	3.52	3.81	5.16	0.28	0.20	0.04
	People					
Permute&Rank	6.26	6.46	7.01	0.01	0.01	0.00
Dependency	3.96	4.32	7.03	0.06	0.06	0.00
Template	5.16	4.62	7.26	0.03	0.06	0.00
Template-S	4.25	4.31	7.04	0.18	0.13	0.00
	Overall					
Permute&Rank	5.25	5.45	5.78	0.05	0.03	0.01
Dependency	3.93	4.28	6.20	0.07	0.06	0.01
Template	4.73	4.46	6.25	0.05	0.05	0.01
Template-S	3.86	4.04	6.03	0.23	0.17	0.02

Table 2: Results for realization

Discussion We have shown that by using speaker and recency constraints in standard algorithms, it is possible to achieve performance gains on the attribute selection 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 (style and priming).

5 Surface Realization Experiments

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

Results Table 2 shows the evaluation results for string-edit distance and string accuracy on the development set with three different attributes sets: **DEV** – attributes selected by the human test; **FB-sf** – attributes generated by the full brevity algorithm with speaker frequency; and **DR-sf** – attributes selected by the Dale & Reiter algorithm with speaker frequency.

For the TUNA realization task (**DEV** attributes), our approaches work better for the furniture domain, where there are fewer attributes, than for the people domain. For the furniture domain, the **Template-S** approach achieves lowest string-edit distance, while for the people domain, the **Dependency** approach achieves lowest string-edit distance. The latter method was submitted for human evaluation.

When we consider the "end-to-end" referring expression generation task (**FB-sf** and **DR-sf** attributes), the best overall performing system is the speaker-based template generator with full-brevity and speaker frequency attribute selection. In terms of generated sentence quality, a preliminary and qualitative analysis shows that the combination **Permute & Rank** and **DR-sf** produces more naturalistic phrases.

Discussion Although the Template-S approach achieves the best string edit distance scores overall, it is not very robust. If no examples were found in the training data neither Template approach will produce no output. (This happens twice for each of the domains on the development data.) The Dependency approach achieves good overall performance with more robustness.

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

6 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.

In addition to individual style and priming, another potentially fruitful area for exploration with TUNA-style tasks is human visual search strategies (Rayner, 1998). We leave this idea for future work.

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

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

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

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

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

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

Robert Dale. 1992. *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes*. MIT Press, Cambridge, MA.

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

S. Kanthak, 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*.

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

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

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

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

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

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