Controlling Redundancy in Referring Expressions

Jette Viethen[1], Robert Dale[1], Emiel Krahmer[2], Mariët Theune[3], Pascal Touset[3]

[1]Centre for Language Technology
Macquarie University
Sydney, Australia
jviethen@ics.mq.edu.au rdale@ics.mq.edu.au

[2]Communication & Cognition
University of Tilburg
Tilburg, The Netherlands
e.j.krahmer@uvt.nl

[3] Human Media Interaction
University of Twente
Enschede, The Netherlands
m.theune@utwente.nl p.b.touset@student.utwente.nl

Abstract

Krahmer et al.'s (2003) graph-based framework provides an elegant and flexible approach to the generation of referring expressions. In this paper, we present the first reported study that systematically investigates how to tune the parameters of the graph-based framework on the basis of a corpus of human-generated descriptions. We focus in particular on replicating the redundant nature of human referring expressions, whereby properties not strictly necessary for identifying a referent are nonetheless included in descriptions. We show how statistics derived from the corpus data can be integrated to boost the framework's performance over a non-stochastic baseline.

1. Introduction

A key problem explored within the natural language generation (NLG) literature is the generation of referring expressions. Over the last two decades, a number of algorithms have been developed for constructing *distinguishing descriptions*, noun phrases that uniquely pick out their intended referents from sets of distractors. Many different aspects of referring expressions have been investigated by the research in this field, including, for example, the construction of the shortest possible descriptions, how to incorporate relations between objects into referring expressions, and how to refer to groups of objects or parts of objects. However, one aspect of human-generated referring expressions that has not been subjected to much exploration in NLG research is the phenomenon of *redundancy*.

Psycholinguistic evidence suggests that humans tend to overspecify when they describe an object: i.e., they often include information that is not strictly necessary in order to distinguish the object from others around it. This phenomenon has been recognised in the psycholinguistics literature for a long time (see e.g., Whitehurst and Sonnenschein (1976), Sonnenschein (1985), Pechmann (1989)), but has only been a focus of discussion in the NLG literature more recently (see e.g., Viethen and Dale (2006)). That it might be appropriate to incorporate redundancy in the machine generation of referring expressions was already noted in the development of Dale and Reiter's (1995) Incremental Algorithm (IA), which has become something of a standard approach to producing overspecified referring expressions. However, the redundancy the IA allows for has no necessary correlation with the kinds of redundancy found in human-generated referring expressions. In particular, the IA does not provide a mechanism fine-grained enough to directly influence which properties can occur redundantly.

In this paper we explore how to use the mechanisms provided by another framework for the generation of referring expressions, the graph-based framework introduced by Krahmer et al. (2003), in order to fine-tune the redundancy that occurs in referring expressions. Here, the search for an appropriate referring expression is cast in terms of a graph representation. The graph-based framework uses two parameters: a cost function over the possible constituents of a referring expression, and the order in which properties

are considered for inclusion in a referring expression. It forms the basis for a number of recent approaches such as (van der Sluis, 2005; van Deemter and Krahmer, 2007; Viethen and Dale, 2007). Interestingly, despite the popularity of the framework, no significant studies have been carried out to determine how the cost function and property ordering can be used to achieve the ideal output for a given domain.

We present a corpus-based approach to setting these two parameters. To this end, we analysed the TUNA corpus of human-produced referring expressions described in Section 4. The approach is centred around the observation that certain properties are overrepresented in the corpus and are used redundantly more often than other properties. We show how cost functions and property orderings can be employed to gear the graph-based algorithm towards including this redundant information in a referring expression.

An automatic evaluation of the produced output of the algorithm was carried out by comparing it to a test set held back from the TUNA corpus, using the DICE coefficient of similarity. From the results of this evaluation it seems clear that cost functions that allow for redundancy in the generated descriptions are advantageous and that leveraging the interplay between cost function and property ordering in the graph-based algorithm further increases human-likeness of the output.

The next section outlines the development of the field of referring expression generation with regard to the question of redundancy. Following this in Section 3, we give an explanation of how the graph-based algorithm works. Section 4 describes the details of the TUNA corpus relevant to our approach. In Section 5 we discuss the cost functions and property orderings used; and in Section 6 we report the evaluation results. Finally, Section 7 summarises the conclusions from this work and points to some future plans.

2. Redundancy in Referring Expressions

Much work in referring expression generation draws on notions from the Gricean Maxims of conversation (Grice, 1975), which lay out a set of rules that a speaker follows in order to efficiently communicate with a hearer. Specifically the Maxim of Quantity, which states that a speaker's contribution should be as informative as required, but no more informative than required, was interpreted in early work strictly to mean that no redundant properties should be included in a referring expression. This resulted in algorithms aiming for descriptions that are as short as possible containing no redundant information at all (Dale, 1992).

However, in recent decades a lot of work in psycholinguistics has brought evidence that speakers do include redundant information in referring expressions, and not simply as a result of some processing error which leads them astray from the Maxim of Quantity. Especially salient properties such as the type and colour of a referred–to object are frequently used even when they are not needed to distinguish the object from distractors around it (Schriefer and Pechmann, 1988; Pechmann, 1989; Eikmeyer and Ahlsén, 1996; Belke and Meyer, 2002). Most recently an off-line study by Engelhardt et al. (2006) shows that neither speakers nor listeners have a problem with redundancy in referring expressions.

Approaches to the automatic generation of referring expressions based on Dale and Reiter's (1995) Incremental Algorithm take this evidence on board, reinterpreting the Gricean Maxim of Quantity by allowing for some redundancy to occur. They assume incremental processing during language production—similar to the incremental process of speech production put forward by Levelt (1989)—for the subtask of content selection for referring expressions: properties are considered for inclusion into the description in a predefined domain-specific order and can not be removed, even if they turn out to be redundant after more properties have been included.

In the Incremental Algorithm, the only way to influence the selection process is via the order in which the properties are being considered. We chose to use the graph-based framework for our explorations which provides cost functions over the properties as a much more fine-grained mechanism to tune the search for an adequate referring expression.

3. The Graph-based Framework

Krahmer et al.'s (2003) graph-based framework reformulates the task of selecting attributes for referring expressions as a graph-theoretical problem. To this end, the physical scene including the target referent and objects around it is represented as a labelled directed graph. The graph representation of such a scene models each object of the scene as a vertex in the *scene graph*. Direct properties such as COLOUR, TYPE or SIZE are represented as looping edges on the corresponding node. They are labelled with the property names and the values the object in question has for these properties. Relations between objects, for example BELOW or INSIDE, are modelled as edges between the corresponding vertices. Figure 1 shows a sample scene graph containing three objects: a dog, a dog house and a tree. Each of these objects has a number of direct properties represented as loops on

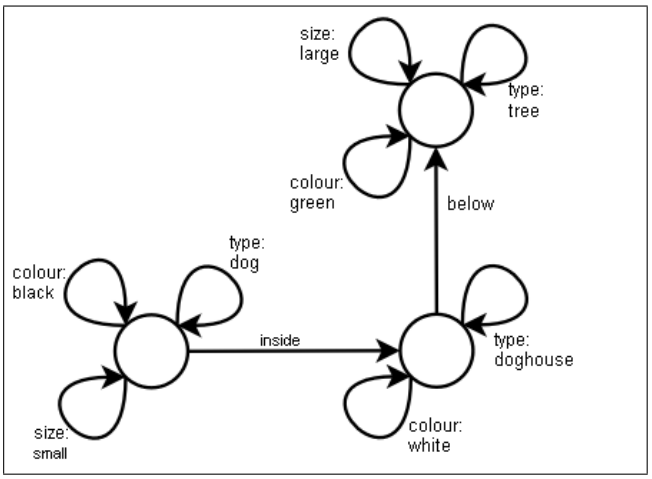

Figure 1: A sample scene graph.

the respective vertex.

To generate a distinguishing description, the graph-based algorithm searches for a subgraph of the scene graph that uniquely refers to the target referent. Starting with the subgraph only containing the vertex which represents the target referent, it performs a depth-first search over the edges connected to the subgraph found so far. It searches the space exhaustively, but uses a cost-based heuristic (described below) to prune the search space. Informally, a subgraph refers to the target object if and only if it can be 'placed over' the domain graph in such a way that the subgraph vertex representing the target object can be 'placed over' the vertex of the target in the domain graph, and each of the labelled edges in the subgraph can be 'placed over' a corresponding edge in the domain graph with the same label. Furthermore, a subgraph is distinguishing if and only if it fits exactly one vertex in the domain graph. The informal notion of one graph being 'placed over' another corresponds with the well-known mathematical graph construction called subgraph isomorphism. An example for a distinguishing subgraph describing the *dog* in our sample domain graph would be [V_1, (V_1 TYPE:DOG V_1), (V_1 INSIDE V_2), V_2, (V_2 TYPE:DOGHOUSE V_2)] which could be realised as "the dog inside the doghouse". Of course, "the dog" or "the small dog" would in this case also suffice as distinguishing descriptions.

3.1. Graphs and Cost Functions

As mentioned above, the graph-based framework uses a cost function to guide the search through the space of possible subgraphs. This cost function is defined over all edge labels and vertices in the domain graph. For our purpose, it is sufficient to assign costs to the edge labels, i.e. the different values each property can take. Because we are not aiming at producing referring expressions containing relations between objects, each subgraph is only going to contain the one vertex representing the referent object. The cost of a subgraph is then defined as the sum over all edges contained in it.

The search algorithm is guaranteed to find the cheapest subgraph representing a distinguishing description for the target referent. To avoid an exhaustive brute-force search through the entire space of subgraphs, the cost function is used as a heuristic to prune a search branch as soon as it becomes as or more expensive than the cheapest distinguishing subgraph found so far.

Using a cost function as a means to indicate preference of certain properties over others enables us to specify the extent of the preference as well as equal preference for certain properties or property values. Let's assume our target object is a *friendly, small, white poodle* and the two cheapest distinguishing descriptions for it are:

1. "The friendly poodle." [POODLE, FRIENDLY]
2. "The small white one." [WHITE, SMALL]

If the property costs are c(POODLE) = 1, c(WHITE) = c(SMALL) = 11, and c(FRIENDLY) = 12, then POODLE is very much preferred over the other properties, and WHITE and SMALL are equally preferred. For this cost function, the algorithm will choose description 1 with cost 13 over description 2 with cost 22, although both properties appearing in description 2, WHITE and SMALL, are preferred over FRIENDLY which appears in description 1. If the cost distribution for POODLE, WHITE, SMALL, and FRIENDLY was 1, 2, 3, and 12 respectively, description 2 with cost 5 would be chosen over description 1 costing 13, although POODLE in description 1 is preferred over both WHITE and SMALL in description 2.

This kind of choice is not possible in approaches that only use a preference ordering over the properties, such as the Incremental Algorithm, as there would be no way to tell whether the preference for POODLE over WHITE and SMALL outweighs the dispreference for FRIENDLY over WHITE and SMALL.

3.2. Graphs and Property Orderings

Of course, it is still possible that more than one distinguishing subgraph with the lowest cost exists in a particular scene graph with a particular cost function. In this case the subgraph encountered first will be the one returned by the algorithm as the description for the target referent. The order in which subgraphs, i.e. descriptions, are found is dependent on the order in which edges, i.e. properties, were considered during

the search process. This means the earlier a property is considered, the more likely it is going to be part of the referring expression produced when there is more than one cheapest solution.

Consider again the case of the *friendly small white poodle* from Section 3.1 with the same two distinguishing descriptions. Let the properties POODLE, WHITE, SMALL, and FRIENDLY have cost 1, 2, 3 and 4 respectively. In this case both descriptions have the same cost (5), which means the cost function cannot help us arbitrate between them. As just mentioned, in such a case the graph-based algorithm chooses the description it encounters first. Which description is encounters first is determined entirely by the order in which it considered the properties. If it started building a description with POODLE or FRIENDLY, it will return description 1 ("The friendly poodle"); if it started with WHITE or SMALL, it will return description 2 ("The small white one").

This might seem like a highly hypothetical example with a very low likelihood of arising. However, as we will see in Section 5.2, in order to allow some properties to occur redundantly, we need to assign them zero cost, which in turn makes solutions with the same cheapest cost more likely.

4. The TUNA Corpus

The data used for training, development and testing is a subset of the Aberdeen TUNA corpus, a human-generated data set designed for the task of selecting properties for referring expressions (Gatt et al., 2007) which was used for the 2007 ASGRE Challenge[1] (Belz and Gatt, 2007).

The data set is divided into a Furniture domain and a People domain, focussing on descriptions of singular objects; it does not contain any spatial relations between objects as part of referring expressions. Human participants in an online data collection experiment saw on the screen a scene containing either a number of furniture items, such as shown in Figure 2, or photographs of people. They were then asked to type a description for a highlighted item, the target referent. The training set used in this study contains 239 trials from the Furniture domain and 206 trials from the People domain. The development and test sets each consist of 80 Furniture trials and 68 People trials.

Each trial in the corpus consists of an XML representation of the attributes of all objects contained in a scene and the attributes contained in a human-produced description for the target referent. The target referent is marked by a special XML attribute tag. Several human descriptions exist for each scene–target pair. Figure 3 shows the XML encoding of a sample description given for the target object in Figure 2, marked by a square around it.

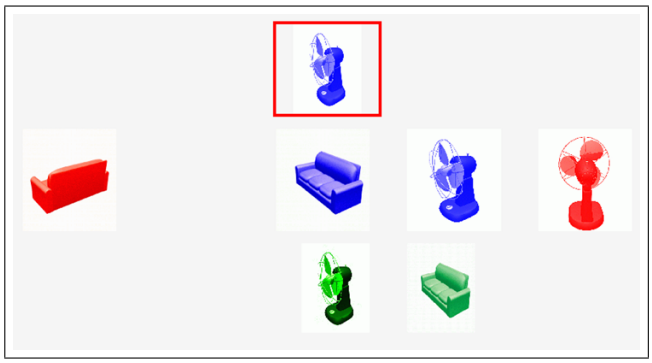

Figure 2: Monochrome picture of a scene from the Furniture domain of the TUNA Corpus.

```
<DESCRIPTION NUM="SINGULAR">
<DET ID="396" VALUE="definite">The </DET>
<ATTRIBUTE ID="a1" NAME="size"
VALUE="small"> small </ATTRIBUTE>
<ATTRIBUTE ID="a2" NAME="colour"
VALUE="blue"> blue </ATTRIBUTE>
<ATTRIBUTE ID="a3" NAME="type"
VALUE="fan"> fan </ATTRIBUTE>.
</DESCRIPTION>
```

Figure 3: Example of the XML encoding of a description in the TUNA Corpus.

Properties that occur in both domains are TYPE, ORIENTATION and X- and Y-DIMENSIONS. Additional attributes in the Furniture domain are COLOUR and SIZE, and additional attributes in the People domain are AGE, ORIENTATION, HAIR COLOUR, and the presence or absence of HAIR, BEARDS, GLASSES and different CLOTHING items.

5. Approach

We experimented with various cost functions, some of which were suggested in Krahmer et al. (2003), and some that we specifically created for this data set. Since people tend to mention the TYPE of an object (typically realized as the head noun) regardless of whether it is a distinguishing property, we modified the graph-based algorithm so that type is selected by default. In the data set, type was mentioned in 98% and 92% of the descriptions for Furniture and People respectively.

[1]For more information on the Attribute Selection for the Generation of Referring Expressions Challenge go to http://www.csd.abdn.ac.uk/research/evaluation/.

5.1. Frequency Counts of Properties

The graph-based framework gives an excellent opportunity to use statistical methods in combination with a rule-based approach to referring expression generation. Therefore, some of our functions use stochastic costs based on the frequency of each property value in the human descriptions from the training set of the TUNA corpus. This is based on the reasoning that choosing properties that occur frequently in the training set should increase the likelihood that the expressions containing these properties match the expressions in the test set.

Based on this rationale, the more frequently the value occurs in this set, the cheaper it is in the stochastic cost functions. Property costs in this function were derived by rounding $-log_2(P(v))$ to the first decimal and multiplying by 10, where $P(v)$ is the probability that property value v (corresponding to an edge in the scene graph) occurs in a description, given that the target object actually has this property. The probability $P(v)$ is estimated by determining the frequency of each property value in the training corpus, relative to the number of target objects that possess this property value.[2]

For reasons of space, Table 1 only shows the combined percentages and average stochastic costs per property type, rather than the costs for each attribute–value pair.

FURNITURE				
Property	**# in desc**	**P(v)**	$-log_2(P(v))$	**cost**
colour	211	0.88	0.18	2
orientation	84	0.35	1.51	15
size	86	0.36	1.47	15
x-Dim	48	0.20	2.32	23
y-Dim	64	0.27	1.90	19

PEOPLE				
Property	**# in desc**	**P(v)**	$-log_2(P(v))$	**cost**
age	12	0.06	4.08	41
orientation	4	0.02	5.97	60
hairColour	68	0.33	1.60	16
hasSuit	3	0.01	6.27	63
hasShirt	2	0.01	6.96	70
hasTie	1	0.00	7.97	80
has Beard	88	0.37	1.44	14
hasGlasses	90	0.38	1.41	14
hasHair	33	0.35	1.51	15
x-Dim	60	0.25	1.99	20
y-Dim	63	0.32	1.66	17

Table 1: Average stochastic costs based on property frequency in the training sets.

5.2. Redundancy in the Graph-based Algorithm

To guarantee that the search process of the graph-based algorithm terminates, the cost function has to be monotonic increasing: i.e. the addition of a new property to a description can never decrease the cost of the description. In general, this monotonicity constraint prevents the graph-based algorithm from including any properties in a referring expression which are not necessary for distinguishing the target referent from the distractor objects around it. As long as each property has a positive cost, and if description A is a subset of description B, then description A is always cheaper than description B. Consequently, if A is already distinguishing, the algorithm returns A and not B.

However, the framework does offer a mechanism which allows us to specify properties that should be used redundantly: introducing properties with cost 0 allows them to occur in a referring expression even if they do not contribute to the distinctiveness of the description. Note that the cost function is not required to be *strictly* monotonic; zero costs are therefore allowed. In this case two distinguishing descriptions that only differ because one of them contains a zero-cost property will have the same cost. Of course, then we have the problem of having to decide which of these two cheapest solutions should be returned. This means that giving properties zero cost does still not guarantee that they will actually be included in the final output: it could still be the other cheapest solution without the redundant property that gets produced.

As discussed in Section 3.2 above, the choice between descriptions with the same cost is determined by the order in which the algorithm considers properties for inclusion into a referring expression. For example, we might want to allow COLOUR to occur redundantly in our descriptions and therefore give it cost 0. If the target object can be distinguished from all other objects by

A) "The front facing desk", or

B) "The *blue* front facing desk",

then (B) will only be returned if COLOUR:BLUE is considered before ORIENTATION:FRONT.

In order to investigate the effect of the property ordering on the performance of the system, we tested

[2] We also computed these values based on the combined training and development sets. However, the cost functions based on these re-computed values did not perform significantly better and in some cases even worse than the ones based only on the training data.

each cost function with two different property orderings. The first lists all properties in a *Random Order* and was used as the baseline; the second lists *Free Properties First*, but keeps the rest of the properties in alphabetical order. *Free Properties First* ensures that zero-cost properties will be included in the referring expression produced, even if their appearance is informationally redundant. Our hypothesis was this property ordering would outperform the first.

5.3. The Cost Functions

We devised four cost functions with which we tested the algorithm:

Simple Costs: In our simplest cost function, all attributes cost 1. Using *Simple Costs*, the graph-based algorithm resembles Dale's (1992) Full Brevity algorithm in that it always favours the shortest distinguishing description. This cost function is used as a baseline for our hypothesis that stochastic costs based on frequency counts from a human-produced corpus would outperform such a Full Brevity account

Stochastic Costs: This cost function assigns each property the frequency-based stochastic cost listed in the last column of Table 1.

Free-Stochastic: introduces zero costs for the properties which are highly over-represented and used redundantly more often than other properties in the human data we analysed. These are COLOUR for the Furniture domain and HASBEARD and HASGLASSES for the People domain. Table 1 shows that these are also overall the most used properties. The other properties are assigned their stochastic costs.

Free-Naïve: This cost function more coarsely translates the frequency counts from Table 1 into property costs. It allows us to test to a certain degree how fine-grained our cost functions need to be. Zero cost is assigned to the same properties as in *Free-Stochastic*, cost 2 to the most rarely used properties (X- and Y-DIMENSIONS in the Furniture domain and HASSUIT, HASSHIRT and HASTIE in the People domain), and cost 1 to all remaining properties.

6. Evaluation

For the evaluation we paired the different cost functions with each of the two property orderings, resulting in 8 different conditions to be assessed against the test sets from each domain held back from the TUNA corpus. The test set for the Furniture domain contained 80, and that for the People domain 68 items. As with the development and training sets, the test sets contain descriptions from several human participants for each scene–target pair. The system output for each such pair was compared to each of the human descriptions using the DICE coefficient of similarity and the results were averaged to obtain an overall score for each domain.

6.1. The DICE Coefficient

The DICE coefficient of similarity provides an estimate of the similarity of two sets of attributes. Let A and B be two such sets of attributes. The DICE score for these two sets is calculated as the number of attributes both sets have in common multiplied by 2 and divided by the sum of the cardinalities of the sets:

$$DICE(A, B) = \frac{2 \times |A \cap B|}{|A \cup B|}$$

DICE scores vary between 0 for attribute sets that have nothing in common and 1 for a perfect match between two sets.

6.2. Results

The DICE scores achieved by the graph-based framework using the cost functions from Section 5.3 and the two tested property orderings are displayed in Tables 2 and 3. For each domain we report the mean DICE score and the perfect recall percentage (PRP), which is the proportion of DICE scores equal to 1. DICE and PRP were the official measures used for the evaluation of attribute selection in the 2007 ASGRE Challenge.

	Furniture		*People*	
Cost function	**Mean**	PRP	**Mean**	PRP
Simple Costs	0.550	2.5	0.606	17.6
Stochastic Costs	0.658	18.8	0.625	17.6
Free-Stochastic	0.701	27.5	**0.665**	16.2
Free-Naïve	0.757	33.8	0.647	**19.1**

Table 2: Results for the test sets with the *Random Order* baseline.

	Furniture		*People*	
Cost function	**Mean**	PRP	**Mean**	PRP
Simple Costs	0.597	12.5	0.569	17.7
Stochastic Costs	0.669	21.3	0.625	17.7
Free-Stochastic	0.775	46.3	**0.689**	**25**
Free-Naïve	**0.796**	**50**	0.639	20.6

Table 3: Results for the test sets with the *Free Properties First* ordering.

Overall, the two tables show that results for the *Free Properties First* ordering are better than those for the *Random Order*. The only exception is the *Simple*

Costs function, which performs better with the *Random Order* in the People domain.

The cost functions based on the frequency counts of properties from the training set perform consistently better than the baseline *Simple Costs*. The only case where the difference between *Simple Costs* and the next lowest scoring *Stochastic Costs* function is not very pronounced is with the *Random Order* in the People domain.

The two cost functions which allow redundancy to occur by setting the cheapest properties to cost zero, *Free-Stochastic* and *Free-Naïve*, score consistently higher than the *Stochastic Costs* function that does not have any zero-cost properties.

Interestingly, *Free-Stochastic* seems to be doing better than *Free-Naïve* on the People domain for both property orderings, while the opposite is true in the Furniture domain.

6.3. Discussion

The general trend of the results from this evaluation clearly suggests three points:

1. Cost functions based on stochastic values clearly outperform a simple baseline resembling many traditional approaches which had the aim of producing the shortest possible description. This demonstrates the validity of the growing trend towards using corpus-based methods in NLG.

2. Cost functions that allow redundancy to occur in the referring expressions generated by the graph-based algorithm better match human referring behaviour than those that don't.

3. Leveraging the interplay between cost function and property ordering to gear the graph-based framework towards producing more overspecified output further increases human-likeness. With the *Free Properties First* ordering our *Free-Naïve* cost function matches the DICE score of the winning peer system from the 2007 ASGRE Challenge in the Furniture domain and beats it on PRP.

The fact that the *Free-Stochastic* cost function outperforms *Free-Naïve* on the People domain for both property orderings, while the opposite is true in the Furniture domain, is most likely due to the stochastic costs in the People domain being spread over a wide range of values, which means that by simplifying the cost function a lot of information about the human preferences for using certain properties is lost.

7. Conclusions and Future Work

The evaluation scheme used in this study tests explicitly for human-likeness of the referring expressions produced. This is based on the assumption that the more human-like a referring expression is, the more natural it sounds. However, no claim can be made that this approach maximises the usefulness of descriptions for a listener in a specific task scenario. As we strive to mimic the data in a corpus, we cannot avoid also mimicking any deficiencies it contains. Furthermore, in using automatic evaluation schemes it is paramount to keep in mind that the metrics and corpora used all have advantages and disadvantages and put emphasis on different aspects of natural language generation. We are therefore planning to extend the evaluation of our system by using other evaluation metrics and corpora as well as conducting task-based human evaluation to assess its usefulness as well as the naturalness.

One drawback of the graph-based framework is its reliance on static, pre-determined cost functions and property orderings. This does not allow us to capture dependencies between the use of properties, such as the fact that HAIRCOLOUR in the People domain only gets used if HASHAIR is also included. In order to be able to model dependencies of this kind, we will need to implement dynamic cost functions and property orderings that make the inclusion of a property more or less likely depending on which properties have already been chosen. Despite this limitation of the framework in its current form, we believe that it is the most promising framework for referring expression generation available to date.

The results reported in this paper demonstrate that using corpus statistics for tuning the parameters of the graph-based framework can boost its performance. To our knowledge, this is the first approach reported in the literature to systematically determine and evaluate settings for both parameters of the graph-based framework. We believe that this research paves the way for a more controlled approach to the generation of redundant descriptions which better match those produced by humans.

8. References

Eva Belke and Antje S. Meyer. 2002. Tracking the time course of multidimensional stimulus discrimination: Analysis of viewing patterns and processing time during same-different decisions. *European Journal of Cognitive Psychology*, 14(2):237–266.

Anja Belz and Albert Gatt. 2007. The attribute selection for GRE challenge: Overview and evaluation results. In *Proceedings of UCNLG+MT: Language*

Generation and Machine Translation, pages 75–83, Copenhagen, Denmark.

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

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

Hans-Jürgen Eikmeyer and Elisabeth Ahlsén. 1996. The cognitive process of referring to an object: A comparative study of German and Swedish. In *Proceedings of the 16th Scandinavian Conference on Linguistics*, Turku, Finland.

Paul E. Engelhardt, Karl D. Bailey, and Fernanda Ferreira. 2006. Do speakers and listeners observe the Gricean maxim of quantity? *Journal of Memory and Language*, 54:554–573.

Albert Gatt, Ielka van der Sluis, and Kees van Deemter. 2007. Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the 11th European Workshop on Natural Language Generation*, pages 49–56, Schloß Dagstuhl, Germany.

H. Paul Grice. 1975. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics Volume 3: Speech Acts*, pages 43–58. Academic Press, New York, NY.

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

Willem M. J. Levelt. 1989. *Speaking: From intention to articulation*. MIT Press, Cambridge, MA.

Thomas Pechmann. 1989. Incremental speech production and referential overspecification. *Linguistics*, 27:89–110.

Herbert J. Schriefer and Thomas Pechmann. 1988. Incremental production of referential noun phrases by human speakers. In Michael Zock and Gérard Sabah, editors, *Advances in Natural Language Generation*, volume 1. Pinter, London, UK.

Susan Sonnenschein. 1985. The development of referential communication skills: Some situations in which speakers give redundant messages. *Journal of Psycholinguistic Research*, 14(5):489–508.

Kees van Deemter and Emiel Krahmer. 2007. Graphs and Booleans: On the generation of referring expressions. In Harry C. Bunt and Reinhard Muskens, editors, *Computing Meaning*, volume 3, pages 17–53. Kluwer, Dordrecht, The Netherlands.

Ielka van der Sluis. 2005. *Multimodal Reference, Studies in Automatic Generation of Multimodal Referring Expressions*. Ph.D. thesis, Tilburg University, Tilburg, The Netherlands.

Jette Viethen and Robert Dale. 2006. Algorithms for generating referring expressions: Do they do what people do? In *Proceedings of the 4th International Conference on Natural Language Generation*, pages 63–70, Sydney, Australia, July.

Jette Viethen and Robert Dale. 2007. Capturing acceptable variation in distinguishing descriptions. In *Proceedings of the 11th European Workshop on Natural Language Generation*, Schloß Dagstuhl, Germany.

Grover J. Whitehurst and Susan Sonnenschein. 1976. The development of communication: Attribute variation leads to contrast failure. *Child Development*, 47:473–482.