# Corpus-based Referring Expressions Generation

**Hilder V. L. Pereira, Eder M. de Novais, André C. Mariotti, Ivandré Paraboni**

School of Arts, Sciences and Humanities, University of São Paulo
Av. Arlindo Bettio, 1000 - São Paulo, Brazil
hilder.pereira@usp.br, eder.novais@usp.br, mariotti5@usp.br, ivandre@usp.br

**Abstract**

In Natural Language Generation, the task of attribute selection (AS) consists of determining the appropriate attribute-value pairs (or semantic properties) that represent the contents of a referring expression. Existing work on AS includes a wide range of algorithmic solutions to the problem, but the recent availability of corpora annotated with referring expressions data suggests that corpus-based AS strategies become possible as well. In this work we tentatively discuss a number of AS strategies using both semantic and surface information obtained from a corpus of this kind. Relying on semantic information, we attempt to learn both global and individual AS strategies that could be applied to a standard AS algorithm in order to generate descriptions found in the corpus. As an alternative, and perhaps less traditional approach, we also use surface information to build statistical language models of the referring expressions that are most likely to occur in the corpus, and let the model probabilities guide attribute selection.

**Keywords:** Text Generation, Referring Expressions, Attribute Selection

## 1. Introduction

In Natural Language Generation (NLG) Systems, Referring Expressions Generation (REG) is the task of providing linguistic forms to describe discourse objects, as in 'the man with a red hat', 'him', 'Mr. Johns', etc. Much of the research in the field has focused on the computation of the semantic contents of referring expressions to be realised as definite descriptions (e.g., 'Please follow *the left corridor*'), known as the attribute selection (AS) task. AS is also the focus of this paper.

Attribute selection consists of determining the appropriate attribute-value pairs (or semantic properties) that represent the contents of a referring expression. The definition of what exactly an 'appropriate' attribute is remains an open research question, but it is generally agreed that these attributes should help ruling out potential distractors (i.e., other objects in the same context as the intended referent) whilst preventing false conversational implicatures in the sense defined by H. P. Grice (Grice, 1975).

Existing work on AS includes a wide range of algorithmic solutions (Dale and Reiter, 1995; Krahmer et al., 2003; Paraboni et al., 2007). In recent years, however, the availability of corpora annotated with referring expressions data such as TUNA (van Deemter et al., 2006; van Deemter et al., 2011) and GRE3D (Viethen and Dale, 2008) suggests that corpus-based AS strategies become possible as well.

In this work we tentatively discuss a number of AS strategies using both semantic and surface information obtained from the TUNA corpus. Relying on semantic information, we attempt to learn both global and individual AS strategies that could be applied to the Incremental algorithm (Dale and Reiter, 1995) in order to generate TUNA descriptions. As an alternative, and perhaps less traditional approach, we also use surface information to build statistical language models of the referring expressions that are most likely to occur in the corpus, and let the model probabilities guide attribute selection using the graph-based approach described in (Krahmer et al., 2003).

## 2. Background

One of the best-known approaches to AS is the Incremental algorithm described in (Dale and Reiter, 1995), which takes as an input a context set C containing domain objects with their corresponding properties, and a list P representing the order in which the attributes will be considered for selection. One particular object in C is the target r to be described by means of a referring expression (i.e., a uniquely identifying set of properties), and the remaining objects are assumed to be distractors. The goal of the algorithm is to compute a list of attributes L such that L uniquely describes r and no other distractor in C.

The Incremental algorithm works by iterating over the list of preferred attributes P and adding to the description L under construction the attributes of the target object r that help ruling out at least one distractor, until the point in which L uniquely describes r. For instance, consider the following domain entities and their properties:

$E_1$ : (type, chair), (size, small), (colour, black)
$E_2$ : (type, chair), (size, large), (colour, white)
$E_3$ : (type, desk), (size, small), (colour, black)
$E_4$ : (type, chair), (size, small), (colour, red)

Example 1 - A context set conveying four entities and their referable attribute-pair values

In this example, assuming the preferred order P = {*type, size, colour*}, we may refer to $E_1$ as L = {*type-chair*, *size-small*, *colour-black*}, which could be realised as 'the small black chair'. The reference to the *size* attribute rules out $E_2$, which is large (i.e., not small), the reference to *type* rules out $E_3$, which is a desk, and the reference to *colour* rules out $E_4$, which is red. Using the same P strategy $E_2$ would be described as 'the large chair', $E_3$ simply as 'the desk' and $E_4$ as 'the small red chair'. Attributes that do not rule out any distractors are best avoided since they may lead to false conversational implicatures (Grice, 1975).

In this work we will consider the problem of generating singular referring expressions in the Furniture domain[1] as seen in the TUNA corpus of referring expressions (van Deemter et al., 2006). The TUNA corpus consists of situations of reference (or trials) collected in a number of controlled experiments for the purpose of REG research. Each TUNA trial comprises a number of referable objects represented by sets of semantic properties, including a target object and its distractors. In addition to that, each trial includes a uniquely identifying description of the target object as produced by a native or fluent speaker of English. TUNA descriptions are represented in the corpus both as a set of semantic properties (or attribute set) and as the actual surface string (or string description) uttered by each participant.

The following is a simplified example of a single entity definition adapted from the TUNA corpus (a typical TUNA trial will contain several such definitions, one for each object in the context):

```
<Entity Id="23" Image="desk.gif" Type="target" >
   <Attr Name="colour" Type="literal" Value="grey" />
   <Attr Name="orient" Type="literal" Value="front" />
   <Attr Name="type" Type="literal" Value="desk" />
   <Attr Name="size" Type="literal" Value="large" />
   <Attr Name="x-dim" Type="gradable" Value="unk" />
   <Attr Name="y-dim" Type="gradable" Value="unk" />
</Entity>
```

The actual description produced in each trial is represented as the following simplified example, also adapted from the TUNA corpus:

```
<String Description> the gray desk </String Description>
<Attribute Set>
   <Attr Id="a2" Name="type" Value="desk" />
   <Attr Id="a1" Name="colour" Value="grey" />
</Attribute Set>
```

Different portions of the TUNA corpus have been extensively used in a series of Shared Tasks (Belz and Gatt, 2007; Gatt et al., 2008; Gatt et al., 2009), including an early version of one of our present baseline systems described in (Lucena and Paraboni, 2008; Lucena et al., 2010). For further details on the TUNA corpus we refer to (Gatt and Belz, 2010) and (van Deemter et al., 2011). In what follows we will use semantic information obtained from the Furniture subcorpus of singular descriptions in two machine learning approaches to AS discussed in Section 3. Word string information is the basis of a shallow AS approach described in Section 4.

## 3. Learning Attribute Selection Strategies from Corpora

Using the input-output pairings (i.e., every context C and description L pairs) provided by the TUNA corpus as discussed in the previous section, we consider two learning methods for AS. In both cases, we take as learning features the set of integer values representing the discriminatory power of each referable attribute (i.e., the number of distractors ruled out by each attribute of the target object, such as *colour*, *size* etc.) For instance, in a context set as seen in previous Example 1, assuming that we would like to refer to $E_4$, the corresponding discriminatory power values would be defined as *d_type*=1, *d_size*=1 and *d_colour*=3.

In the first learning method, we will attempt to learn possible $P_j$ orderings (defined as nominal values) that, if applied to the Incremental algorithm, would produce the desired output L for each given input C. In the second method, we will use the input C to (binary) decide whether to select each attribute individually. We will call these our *Global* and *Individual* AS classifiers, which are discussed separately below.

### 3.1. Global AS classification

In (Dale and Reiter, 1995), the preferred order P in which attributes are considered for selection will determine the actual AS strategy of the algorithm, and by varying the order in P it is possible to produce a wide range of alternative descriptions of the target object. For instance, suppose that we use the preferred order P = {*colour*, *type*, *size*} in the context illustrated in previous Example 1. In this case, a reference to $E_3$ would be described as 'the black desk' and not simple as 'the desk' as the preferred order P = {*type*, *size*, *colour*} would obtain.

Assuming that we would like to generate descriptions using the Incremental algorithm, we will use standard learning techniques to discover possible $P_j$ strategies that would produce the same results as those seen in the corpus. To this end, we exhaustively run the Incremental algorithm for each situation of reference (i.e., a target object and distractors as seen in the training corpus), attempting every possible $P_j$ strategy (i.e., every permutation of the target attributes) that (a) starts with *type*[2] and (b) produces the desired output For instance, the possible strategies that would successfully produce, e.g., 'the white chair' as a description of $E_2$ in Example 1 are $P_1$ = {*type*, *colour*, *size*} and $P_2$ = {*colour*, *type*, *size*}, but since we assume *type* to have a fixed position, only $P_1$ needs to be considered.

If a strategy $P_j$ produces the expected output (i.e., the same uniquely distinguishing description found in the corpus) then a training instance is created, in which the learning attributes are the discriminatory power values of the context objects, and the (nominal) class value to be learned is the strategy $P_j$. Finally, given that a single referring expression may be produced by multiple $P_j$ strategies, we keep only the most frequent strategy applicable to each input.

The following Table 1 shows four learning instances for the context in Example 1 using arbitrary descriptions L as examples of the kind of output string found in the TUNA corpus. The learning attributes $d\_type$, $d\_size$ and $d\_colour$ represent discriminatory power values as discussed above. The class to be learned is a nominal value $P_j$ representing the most frequent attribute ordering (as seen in the corpus) that would produce the desired description if used as the P parameter in the Incremental algorithm. For instance, 'tsc'

[1]Available from www.csd.abdn.ac.uk/ agatt/tuna/corpus/

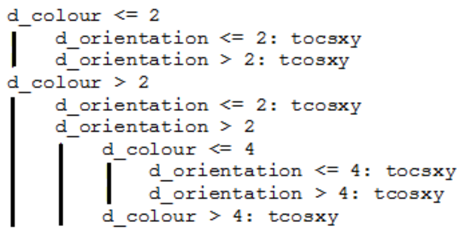
[2]Keeping the type attribute in fixed (i.e., first) position means that *type* is always included in the output description as in (Dale and Reiter, 1995), and this helps reduce the number of permutations to be considered.

stands for P ={*type*, *size*, *colour*}. In our training data we identified 120 distinct AS strategies, although nearly a third of them are highly infrequent, occuring less than 10 times each. The expected output descriptions are included in Table 1 for illustration purposes only, and they are not learning attributes.

| | *Example* | $d\_type$ | $d\_size$ | $d\_colour$ | $P_j$ |
|---|---|---|---|---|---|
| $E_1$ | small black chair | 1 | 1 | 2 | tsc |
| $E_2$ | white chair | 1 | 3 | 3 | tcs |
| $E_3$ | small desk | 3 | 1 | 2 | tsc |
| $E_4$ | red chair | 1 | 1 | 3 | tcs |

Table 1: Examples of learning instances for Global AS classification considering the context in previous Example 1.

Using a training set as above, we applied J48 decision tree-induction (Witten and Frank, 2005) to build a model that, given a context of reference, selects an appropriate $P_j$ strategy for each input. However, given that the choice for one attribute may rule out others, we notice that this is not expected to be a highly accurate model of AS, but simply an estimate of which initial P strategy to use in each case, with average Precision = 0.562, Recall = 0.989 and F-measure = 0.715 using 10-fold cross-validation. Figure 1 shows the pruned tree for the most frequent AS strategies, which includes attributes that we have not considered in the previous examples.



```
d_colour <= 2
|   d_orientation <= 2: tocsxy
|   d_orientation > 2: tcosxy
d_colour > 2
|   d_orientation <= 2: tcosxy
|   d_orientation > 2
|   |   d_colour <= 4
|   |   |   d_orientation <= 4: tocsxy
|   |   |   d_orientation > 4: tcosxy
|   |   d_colour > 4: tcosxy
```

Figure 1: Global AS classifier

Given a target object r and the set V of discriminatory power values of the input context C, a description of r is generated by submitting the input information to the model, and then using the resulting class $P_j$ as the P parameter in the Incremental algorithm, which in turn generates the output description L. This procedure can be summarised as follows.

```
Input:
r : the target object
W : the set of discriminatory power values in C
C : the context set

P ← GlobalClassifier(W)
L ← MakeReferringExpression(r,C,P)
return(L)
```

The algorithm consists of making a call to the decision tree model to obtain the strategy P applied to the Incremental algorithm. The results obtained by this method are discussed in Section 5.

## 3.2. Individual AS classification

As an alternative to the Global AS classifier, we also attempt to learn whether to select each attribute individually as follows. For each TUNA trial taken from a training portion of the corpus, we generate one training instance for every referable attribute $a_j$ except type, thus creating five training instances for each TUNA trial. As in the previous approach, the learning attributes represent the discriminatory power of each attribute of the target object. In the present method, however, the class to be learned is defined as a binary condition representing whether each $a_j$ attribute (*colour*, *size*, *orientation*, *x-dimention* and *y-dimension*) was actually selected to appear in the output description.

Table 2 shows examples of learning classes for the example descriptions in previous Table 1 ($s\_colour$ represents whether the *colour* attribute should be selected etc.) The last three classes have all their values set to zero because these attributes were not used in the previous examples. For the learning attributes under consideration, we refer to previous Table 1.

| *Ex.* | $s\_colour$ | $s\_size$ | $s\_orient$ | $s\_x$ | $s\_y$ |
|---|---|---|---|---|---|
| $E_1$ | 1 | 1 | 0 | 0 | 0 |
| $E_2$ | 1 | 0 | 0 | 0 | 0 |
| $E_3$ | 0 | 1 | 0 | 0 | 0 |
| $E_4$ | 1 | 0 | 0 | 0 | 0 |

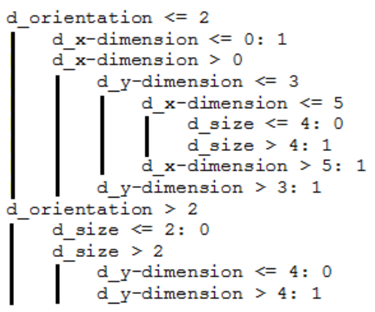
Table 2: Examples of learning classes for Individual AS classification. The learning attributes are the same from Global AS classification ($d\_type$, $d\_size$ and $d\_colour$ ).

From this data set we induced five J48 classifiers using 10-fold cross-validation to determine whether to select colour, size, orientation, x-dimension and y-dimension attributes independently. However, this method was only successful for the selection of size and orientation attributes. These results are shown in Table 3.

| Class | Precision | Recall | F-measure |
|---|---|---|---|
| Colour | 0.643 | 0.514 | 0.500 |
| Orientation | 0.784 | 0.806 | 0.790 |
| Size | 0.886 | 0.895 | 0.891 |
| X-dimension | 0.621 | 0.516 | 0.501 |
| Y-dimension | 0.640 | 0.558 | 0.561 |

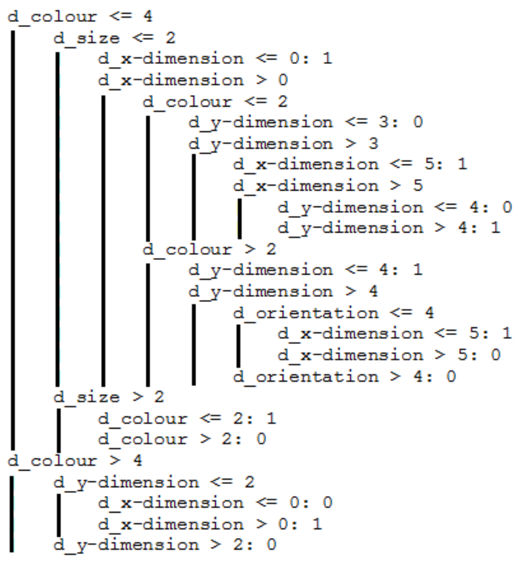
Table 3: Individual AS classification

The induced trees for *size* and *orientation* attribute selection are shown in Figure 2 and Figure 3.



```
d_orientation <= 2
    d_x-dimension <= 0: 1
    d_x-dimension > 0
        d_y-dimension <= 3
            d_x-dimension <= 5
                d_size <= 4: 0
                d_size > 4: 1
            d_x-dimension > 5: 1
        d_y-dimension > 3: 1
d_orientation > 2
    d_size <= 2: 0
    d_size > 2
        d_y-dimension <= 4: 0
        d_y-dimension > 4: 1
```

Figure 2: Individual *size* AS classifier



```
d_colour <= 4
    d_size <= 2
        d_x-dimension <= 0: 1
        d_x-dimension > 0
            d_colour <= 2
                d_y-dimension <= 3: 0
                d_y-dimension > 3
                    d_x-dimension <= 5: 1
                    d_x-dimension > 5
                        d_y-dimension <= 4: 0
                        d_y-dimension > 4: 1
            d_colour > 2
                d_y-dimension <= 4: 1
                d_y-dimension > 4
                    d_orientation <= 4
                        d_x-dimension <= 5: 1
                        d_x-dimension > 5: 0
                    d_orientation > 4: 0
    d_size > 2
        d_colour <= 2: 1
        d_colour > 2: 0
d_colour > 4
    d_y-dimension <= 2
        d_x-dimension <= 0: 0
        d_x-dimension > 0: 1
    d_y-dimension > 2: 0
```

Figure 3: Individual *orientation* AS classifier

The use of the individual classifiers as part of a simple REG algorithm is illustrated as follows, in which The *RulesOut()* function is taken to return the set of objects for which the given property is true.

*Input:*
**r** : the target object
**P** : the list of possible attributes, sorted by frequency
**W** : the set of discriminatory power values in C
**C** : the context set

```
$L \leftarrow \emptyset$
for each $a_i \in P$ do
        if( IndividualClassifier($a_i$ , W) == true )
                $P \leftarrow P - a_i$
                $L \leftarrow L \cup (a_i, v_i)$
                $C \leftarrow C-$ RulesOut $(a_i, v_i)$
repeat
$if(C == \emptyset)$ return(L)
for each $a_i \in P$ do
        if(RulesOut$(a_i, v_i) \neq \emptyset$)
                $L \leftarrow L \cup (a_i, v_i)$
                $C \leftarrow C-$ RulesOut $(a_i, v_i)$
                $if(C == \emptyset)$ return(L)
repeat
return(L)
```

The algorithm starts by including *all* attributes suggested by the binary classifiers in the description L under construction. If L still remains ambiguous after all classifiers have been attempted, the algorithm iterates over a fixed P preference order based on frequency. The list P starts with the attribute *type*, which has the highest frequency in the corpus, and which is always included in the description, as in (Dale and Reiter, 1995). Additional attributes are selected (in this case against the classifiers advice) until L becomes uniquely distinguishing, or until there is no more properties to be considered in P. The results of this method are also discussed in Section 5.

## 4. Word-based Attribute Selection

Our previous approaches take full advantage of the semantic annotation provided by the TUNA corpus to learn possible AS strategies. However, since this kind of information is unlikely to be available in other domains, it is tempting to ask whether we can extract AS strategies directly from word strings. Thus, in a third corpus-based approach to REG, we attempt to use surface information taken from the corpus to guide the (otherwise strictly semantic) AS task.
The word-based approach consists of a standard implementation of the Graph-based algorithm described in (Krahmer et al., 2003) using a modified cost function based on language model probabilities. Briefly, this algorithm takes as an input a context graph V in which nodes represent entities and the edges are properties (self-loops are atomic properties and edges linking two nodes are relational properties.) The goal of the Graph algorithm is to build a sub-graph H that represents a uniquely distinguishing description of the target object. This is achieved by tentatively adding each possible attribute (or graph edge) to the sub-graph under construction with lowest possible cost, which is determined by a customisable cost function. For details regarding the Graph-based algorithm, see (Krahmer et al., 2003).
We use the Graph-based algorithm with a cost function defined by word string probabilities obtained from a language model as follows. First, the word strings in the training portion of the corpus were translated into our target language (Portuguese) and a trigram language model of the target expressions was created. The use of the language model in the cost function of the REG algorithm is illustrated below. The functions *addEdge()* and *removeEdge()* are standard graph methods, the function *Realise()* is taken to call an auxiliary surface realisation engine, and the function *perplexity()* evaluates the language model for the given word string.

*Input:*
**a**$_j$ : the candidate attribute under consideration
**L** : the graph under construction
**M** : the language model

```
addEdge(L,$a_j$)
$s_j \leftarrow$ Realise(L)
$p \leftarrow$ perplexity(M,$s_j$)
removeEdge(L,$a_j$)
return(p)
```

Each candidate attribute $a_j$ is provisionally added to the description L under construction at the beginning of the algorithm, and removed once again when the algorithm terminates. At each step, the description L under consideration is realised as a word string $s_j$ using a simple template-based surface realisation engine of Portuguese definite descriptions. Next, the language model is evaluated against $s_j$, and the model perplexity value p is returned to the main (Graph-based) REG algorithm. The algorithm will then take p as the cost of adding $a_j$ to the description L, and ultimately decide whether to select $a_j$ permanently or not. Thus, we use word string probabilities to predict which semantic properties should be selected in the REG task. Interestingly, although the focus of our work is the AS strategy, this approach produces not only an output attribute set, but also a valid surface realisation form as a by-product. These results are presented in the next section.

## 5. Evaluation

For evaluation purposes, the three corpus-based methods are compared to two baseline systems. The first baseline is a standard implementation of the Graph algorithm presented in (Krahmer et al., 2003), in which we defined a cost function based on the frequency of each attribute in the corpus. In this case, the most frequent attribute (which in the TUNA Furniture domain happens to be *type*) is assigned the lowest cost. The second baseline is the heuristic-based algorithm discussed in (Lucena et al., 2010; Lucena and Paraboni, 2008), which was developed specifically to mirror the descriptions found in the TUNA corpus.

The results are summarised in Table 4 as both Dice and MASI set similarity scores (sorted by MASI means), both of which have been often applied to the evaluation of REG algorithms in the TUNA corpus (van Deemter et al., 2011). For a discussion on the evaluation of REG algorithms using these and other - intrinsic and extrinsic - metrics, see also (Gatt and Belz, 2010).

| **System** | **Dice** | | **MASI** | |
|---|---|---|---|---|
| | Mean | SD | Mean | SD |
| Word-based | 0.64 | 0.2 | 0.26 | 0.2 |
| Frequency-based | 0.53 | 0.3 | 0.32 | 0.3 |
| Global Classifier | 0.69 | 0.3 | 0.44 | 0.3 |
| Individual Classifier | 0.78 | 0.2 | 0.52 | 0.4 |
| Heuristic-based | 0.78 | 0.2 | 0.55 | 0.4 |

Table 4: Results

We performed two separate one-way ANOVAs on Dice and MASI scores, in both cases followed by Tukey HSD test ($\alpha = 0.01$). We found significant differences between the systems in both Dice (F(4,1323)=63.45, MSE=0.068, p<0.01) and MASI scores (F(4,1323)=29.3, MSE=0.136, p<0.01). The homogeneous subsets found are shown in Table 5.

Both Individual and Global classifiers are statistically similar to the Heuristic-based approach. These scores are however lower than those presented in the 2008 TUNA Shared Task (Gatt et al., 2008), in which 14 systems reported Dice scores ranging from 0.23 to 0.86, and MASI scores ranging from 0.09 to 0.70 for the Furniture data. However, as our

| System | | | |
|---|---|---|---|
| Word-based | A | | |
| Frequency-based | A | B | |
| Global Classifier | | B | C |
| Individual Classifier | | | C |
| Heuristic-based | | | C |

Table 5: Homogeneous subsets for Dice and MASI scores. Systems which do not share a letter are significantly different at $\alpha = 0.01$.

test data set the is not the same used in (Gatt et al., 2008), the comparison is not straightforward. The Heuristic-based system, for instance (previously called USP-EACH-FREQ) reported Dice = 0.82 and MASI = 0.62 in (Gatt et al., 2008).

## 6. Final Remarks

This paper presented a number of AS strategies using both semantic and surface information obtained from a corpus of referring expressions. Although the corpus-based approaches do not generally outperform the tailor-made algorithm in (Lucena et al., 2010), we notice that these preliminary results may still be considered satisfactory given that our approaches are in principle adaptable to other domains, whereas the work in (Lucena et al., 2010) remains largely domain-specific. Regarding the difference between our approaches, we notice that even though the learning approach outperforms word-base REG, it still requires a semantically annotated corpus, which may not be easily obtainable for real-world applications in general.

The word-based approach, by contrast, requires only a corpus of word strings. As future work we intend to refine both the learning strategies and the statistical language models used by each method to improve their results, and extend the evaluation to different domains, e.g., by considering the People domain in TUNA (van Deemter et al., 2006).

## 7. Acknowledgements

This work has been supported by FAPESP grant nr. 2009/08499-9 and nr. 2009/09061-7, and by the University of São Paulo.

## 8. References

A. Belz and A. Gatt. 2007. The attribute selection for gre challenge: Overview and evaluation results. In *Proceedings of UCNLG+MT: Language Generation and Machine Translation*.

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

A. Gatt and A. Belz. 2010. Introducing shared tasks to nlg: The tuna shared task evaluation challenges. *Springer Lecture Notes in Artificial Intelligence*, 5790:264–293.

A. Gatt, A. Belz, and E. Kow. 2008. The tuna challenge 2008: Overview and evaluation results. In *Proceedings of the Fifth International Natural Language Generation Conference (INLG-2008)*, pages 198–206.

A. Gatt, A. Belz, and E. Kow. 2009. The tuna-reg challenge 2009: Overview and evaluation results. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 174–182.

H. P. Grice, 1975. *Logic and conversation*, pages 41–58.

E. Krahmer, S. van Erk, and A. Verleg. 2003. Graph based generation of referring expressions. *Computational Linguistics*, 29(1):53–72.

D. J. Lucena and Ivandré Paraboni. 2008. Combining frequent and discriminating attributes in the generation of definite descriptions. *Springer Lecture Notes in Artificial Intelligence*, 5290:252–261.

D. J. Lucena, D. B. Pereira, and Ivandré Paraboni. 2010. From semantic properties to surface text: the generation of domain object descriptions. *Inteligencia Artificial*, 14(45):48–58.

I. Paraboni, K. van Deemter, and J. Masthoff. 2007. Generating referring expressions: Making referents easy to identify. *Computational Linguistics*, 33(2):229–254.

K. van Deemter, I. van der Sluis, and A. Gatt. 2006. Building a semantically transparent corpus for the generation of referring expressions. In *Proceedings of the International Natural Language Generation Conference (INLG-2006)*.

K. van Deemter, A. Gatt, I. van der Sluis, and R. Power. 2011. Generation of referring expressions: Assessing the incremental algorithm. *Cognitive Science*, pages 1–38.

J. Viethen and R. Dale. 2008. The use of spatial relations in referring expressions. In *Proceedings of the 5th International Conference on Natural Language Generation*, pages 59–67, Salt Fork, OH.

I. H. Witten and E. Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufman Publishers, San Francisco, CA, 2nd edition.