Generating Referring Expressions in Open Domains

Advaith Siddharthan
Computer Science Department
Columbia University
as372@cs.columbia.edu

Ann Copestake
Computer Laboratory
University of Cambridge
aac10@cl.cam.ac.uk

Abstract

We present an algorithm for generating referring expressions in open domains. Existing algorithms work at the semantic level and assume the availability of a classification for attributes, which is only feasible for restricted domains. Our alternative works at the realisation level, relies on WordNet synonym and antonym sets, and gives equivalent results on the examples cited in the literature and improved results for examples that prior approaches cannot handle. We believe that ours is also the first algorithm that allows for the incremental incorporation of relations. We present a novel corpus-evaluation using referring expressions from the Penn Wall Street Journal Treebank.

1 Introduction

Referring expression generation has historically been treated as a part of the wider issue of generating text from an underlying semantic representation. The task has therefore traditionally been approached at the semantic level. Entities in the real world are logically represented; for example (ignoring quantifiers), a *big brown dog* might be represented as $\text{big1}(x) \wedge \text{brown1}(x) \wedge \text{dog1}(x)$, where the predicates big1, brown1 and dog1 represent different attributes of the variable (entity) x. The task of referring expression generation has traditionally been framed as the identification of the shortest logical description for the referent entity that differentiates it from all other entities in the discourse domain. For example, if there were a *small brown dog* ($\text{small1}(x) \wedge \text{brown1}(x) \wedge \text{dog1}(x)$) in context, the minimal description for the *big brown dog* would be $\text{big1}(x) \wedge \text{dog1}(x)$[1].

This semantic framework makes it difficult to apply existing referring expression generation algorithms to the many regeneration tasks that are important today; for example, summarisation, open-ended question answering and text simplification. Unlike in traditional generation, the starting point in these tasks is unrestricted text, rather than a semantic representation of a small domain. It is difficult to extract the required semantics from unrestricted text (this task would require sense disambiguation, among other issues) and even harder to construct a classification for the extracted predicates in the manner that existing approaches require (cf., §2).

In this paper, we present an algorithm for generating referring expressions in open domains. We discuss the literature and detail the problems in applying existing approaches to reference generation to open domains in §2. We then present our approach in §3, contrasting it with existing approaches. We extend our approach to handle relations in §3.3 and present a novel corpus-based evaluation on the Penn WSJ Treebank in §4.

2 Overview of Prior Approaches

The *incremental algorithm* (Reiter and Dale, 1992) is the most widely discussed attribute selection algorithm. It takes as input the intended referent and a *contrast set* of *distractors* (other entities that could be confused with the intended referent). Entities are represented as attribute value matrices (AVMs). The algorithm also takes as input a **preferred-attributes** list that contains, in order of preference, the attributes that human writers use to reference objects. For example, the preference might be $\{\texttt{colour}, \texttt{size}, \texttt{shape}...\}$. The algorithm then repeatedly selects attributes from **preferred-attributes** that rule out at least one entity in the contrast set until all distractors have been ruled out.

It is instructive to look at how the incremental algorithm works. Consider an example where a *large brown dog* needs to be referred to. The contrast set contains a *large black dog*. These are represented by the AVMs shown below.

$$\begin{bmatrix} \texttt{type} & \textit{dog} \\ \texttt{size} & \textit{large} \\ \texttt{colour} & \textit{brown} \end{bmatrix} \quad \begin{bmatrix} \texttt{type} & \textit{dog} \\ \texttt{size} & \textit{large} \\ \texttt{colour} & \textit{black} \end{bmatrix}$$

Assuming that the **preferred-attributes** list is $[\texttt{size}, \texttt{colour}, ...]$, the algorithm would first compare the values of the `size` attribute (both *large*),

[1] The predicate dog1 is selected because it has a distinguished status, referred to as *type* in Reiter and Dale (1992). One such predicate has to to be present in the description.

disregard that attribute as not being discriminating, compare the values of the `colour` attribute and return *the brown dog*.

Subsequent work on referring expression generation has expanded the logical framework to allow reference by negation (*the dog that is not black*) and references to multiple entities (*the brown or black dogs*) (van Deemter, 2002), explored different search algorithms for finding the minimal description (e.g., Horacek (2003)) and offered different representation frameworks like graph theory (Krahmer et al., 2003) as alternatives to AVMs. However, all these approaches are based on very similar formalisations of the problem, and all make the following assumptions:

1. A semantic representation exists.
2. A classifi cation scheme for attributes exists.
3. The linguistic realisations are unambiguous.
4. Attributes cannot be reference modifying.

All these assumptions are violated when we move from generation in a very restricted domain to regeneration in an open domain. In regeneration tasks such as summarisation, open-ended question answering and text simplification, AVMs for entities are typically constructed from noun phrases, with the head noun as the type and pre-modifiers as attributes. Converting words into semantic labels would involve sense disambiguation, adding to the cost and complexity of the analysis module. Also, attribute classification is a hard problem and there is no existing classification scheme that can be used for open domains like newswire; for example, WordNet (Miller et al., 1993) organises adjectives as concepts that are related by the non-hierarchical relations of synonymy and antonymy (unlike nouns that are related through hierarchical links such as hyponymy, hypernymy and metonymy). In addition, selecting attributes at the semantic level is risky because their linguistic realisation might be ambiguous and many common adjectives are polysemous (cf., example 1 in §3.1). Reference modification, which has not been considered in the referring expression generation literature, raises further issues; for example, referring to an *alleged murderer* as *the murderer* is potentially libellous.

In addition to the above, there is the issue of overlap between values of attributes. The case of subsumption (for example, that the `colour` *red* subsumes *crimson* and the `type` *dog* subsumes *chihuahua*) has received formal treatment in the literature; Dale and Reiter (1995) provide a find-best-value function that evaluates tree-like hierarchies of values. As mentioned earlier, such hierarchical knowledge bases do not exist for open domains. Further, a treatment of subsumption is insufficient, and degrees of intersection between attribute values also require consideration. van Deemter (2000) discusses the generation of *vague* descriptions when entities have gradable attributes like size; for example, in a domain with four mice sized 2, 5, 7 and 10cm, it is possible to refer to the large mouse (the mouse sized 10cm) or the two small mice (the mice sized 2 and 5cm). However, when applying referring expression generation to regeneration tasks where the representation of entities is derived from text rather than a knowledge base, we have to consider the case where the grading of attributes is not explicit. For example, we might need to compare the attribute *dark* with *black*, *light* or *white*.

In contrast to previous approaches, our algorithm works at the level of words, not semantic labels, and measures the relatedness of adjectives (lexicalised attributes) using the lexical knowledge base WordNet rather than a semantic classification. Our approach also addresses the issue of comparing intersective attributes that are not explicitly graded, by making novel use of the synonymy and antonymy links in WordNet. Further, it treats discriminating power as only one criteria for selecting attributes and allows for the easy incorporation of other considerations such as reference modification (§5).

3 The Lexicalised Approach

3.1 Quantifying Discriminating Power

We define the following three quotients.

Similarity Quotient (SQ)

We define *similarity* as transitive synonymy. The idea is that if X is a synonym of Y and Y is a synonym of Z, then X is likely to be similar to Z. The degree of similarity between two adjectives depends on how many steps must be made through WordNet synonymy lists to get from one to the other.

Suppose we need to find a referring expression for e_0. For each adjective a_j describing e_0, we calculate a similarity quotient SQ_j by initialising it to 0, forming a set of WordNet synonyms S_1 of a_j, forming a synonymy set S_2 containing all the WordNet synonyms of all the adjectives in S_1 and forming S_3 from S_2 similarly. Now for each adjective describing any distractor, we increment SQ_j by 4 if it is present in S_1, by 2 if it is present in S_2, and by 1 if it is present in S_3. SQ_j now measures how similar a_j is to other adjectives describing distractors.

Contrastive Quotient (CQ)

Similarly, we define *contrastive* in terms of antonymy relationships. We form the set C_1 of strict WordNet antonyms of a_j. The set C_2 consists of strict WordNet antonyms of members of S_1

and WordNet synonyms of members of C_1. C_3 is similarly constructed from S_2 and C_2. We now initialise CQ_j to zero and for each adjective describing each distractor, we add $w = \in \{4, 2, 1\}$ to CQ_j, depending on whether it is a member of C_1, C_2 or C_3. CQ_j now measures how contrasting a_j is to other adjectives describing distractors.

Discriminating Quotient (DQ)

An attribute that has a high value of SQ has bad discriminating power. An attribute that has a high value of CQ has good discriminating power. We can now define the Discriminating Quotient (DQ) as $DQ = CQ - SQ$. We now have an order (decreasing DQs) in which to incorporate attributes. This constitutes our **preferred** list. We illustrate the benefits of our approach with two examples.

Example 1: The Importance of Lexicalisation

Previous referring expression generation algorithms ignore the issue of realising the logical description for the referent. The semantic labels are chosen such that they have a direct correspondence with their linguistic realisation and the realisation is thus considered trivial. Ambiguity and syntactically optional arguments are ignored. To illustrate one problem this causes, consider the two entities below:

e1		e2	
type	*president*	type	*president*
age	*old*	age	*young*
tenure	*current*	tenure	*past*

If we followed the strict typing system used by previous algorithms, with **preferred**={age, tenure}, to refer to e1 we would compare the age attributes and rule out e2 and generate *the old president*. This expression is ambiguous since *old* can also mean *previous*. Models that select attributes at the semantic level will run into trouble when their linguistic realisations are ambiguous. In contrast, our algorithm, given flattened attribute lists:

e1		e2	
head	*president*	head	*president*
attrib	*old, current*	attrib	*young, past*

successfully picks *the current president* as *current* has a higher DQ (2) than *old* (0):

attribute	distractor	CQ	SQ	DQ
old	e2{young, past}	4	4	0
current	e2{young, past}	2	0	2

In this example, *old* is a WordNet antonym of *young* and a WordNet synonym of *past*. *Current* is a WordNet synonym of *present*, which is a WordNet antonym of *past*. Note that WordNet synonym and antonym links capture the implicit gradation in the lexicalised values of the age and tenure attributes.

Example 2: Naive Incrementality

To illustrate another problem with the original incremental algorithm, consider three dogs: e1(*a big black dog*), e2(*a small black dog*) and e3(*a tiny white dog*).

Consider using the original incremental algorithm to refer to e1 with **preferred**={colour, size}. The colour attribute *black* rules out e3. We then we have to select the size attribute *big* as well to rule out e2, thus generating the sub-optimal expression *the big black dog*. Here, the use of a predetermined **preferred** list fails to capture what is obvious from the context: that e1 stands out not because it is black, but because it is big.

In our approach, for each of e1's attributes, we calculate DQ with respect to e2 and e3:

attribute	distractor	CQ	SQ	DQ
big	e2{small, black}	4	0	4
big	e3{tiny, white}	2	0	2
black	e2{small, black}	1	4	-3
black	e3{tiny, white}	2	1	1

Overall, *big* has a higher discriminating power (6) than *black* (-2) and rules out both e2 and e3. We therefore generate *the big dog*. Our incremental approach thus manages to select the attribute that stands out in context. This is because we construct the **preferred** list *after* observing the context. We discuss this issue further in the next section. Note again that WordNet antonym and synonym links capture the gradation in the lexicalised size and colour attributes. However, this only works where the gradation is along one axis; in particular, this approach will not work for colours in general, and cannot be used to deduce the relative similarity between yellow and orange as compared to, say, yellow and blue.

3.2 Justifying our Algorithm

The psycholinguistic justification for the incremental algorithm (IA) hinges on two premises:

1. Humans build referring expressions incrementally.
2. There is a preferred order in which humans select attributes (e.g., colour>shape>size...).

Our algorithm is also incremental. However, it departs significantly from premise 2. We assume that speakers pick out attributes that are distinctive *in context* (cf., example 2, previous section). Averaged over contexts, some attributes have more discriminating power than others (largely because of the way we visualise entities) and premise 2 is an approximation to our approach.

We now quantify the extra effort we are making to identify attributes that "stand out" *in a given context*. Let N be the maximum number of entities in

the contrast set and n be the maximum number of attributes per entity. The table below compares the computational complexity of an optimal algorithm (such as Reiter (1990)), our algorithm and the IA.

Incremental Algo	Our Algorithm	Optimal Algo
$O(nN)$	$O(n^2N)$	$O(n2^N)$

Both the IA and our algorithm are linear in the number of entities N. This is because neither algorithm allows backtracking; an attribute, once selected, cannot be discarded. In contrast, an optimal search requires $O(2^N)$ comparisons. As our algorithm compares each attribute of the discourse referent to every attribute of every distractor, it is quadratic in n. The IA compares each attribute of the discourse referent to only one attribute per distractor and is linear in n. Note, however, that values for n of over 4 are rare.

3.3 Relations

Semantically, *attributes* describe an entity (e.g., *the small grey dog*) and *relations* relate an entity to other entities (e.g., *the dog in the bin*). Relations are troublesome because in relating an entity e_o to e_1, we need to recursively generate a referring expression for e_1. The IA does not consider relations and the referring expression is constructed out of attributes alone. The Dale and Haddock (1991) algorithm allows for relational descriptions but involves exponential global search, or a greedy search approximation. To incorporate relational descriptions in the incremental framework would require a classification system which somehow takes into account the relations themselves and the secondary entities e_1 etc. This again suggests that the existing algorithms force the incrementality at the wrong stage in the generation process. Our approach computes the order in which attributes are incorporated *after* observing the context, by quantifying their utility through the quotient DQ. This makes it easy for us to extend our algorithm to handle relations, because we can compute DQ for relations in much the same way as we did for attributes.We illustrate this for prepositions.

3.4 Calculating DQ for Relations

Suppose the referent entity e_{ref} contains a relation $[prep_o\ e_o]$ that we need to calculate the three quotients for (cf., figure 1 for representation of relations in AVMs). We consider each entity e_i in the contrast set for e_{ref} in turn. If e_i does not have a $prep_o$ relation then the relation is useful and we increment CQ by 4. If e_i has a $prep_o$ relation then two cases arise. If the object of e_i's $prep_o$ relation is e_o then we increment SQ by 4. If it is not e_o, the relation is useful and we increment CQ by 4. This is an efficient non-recursive way of computing the quotients CQ and SQ for relations. We now discuss how to calculate DQ. For attributes, we defined $DQ = CQ - SQ$. However, as the linguistic realisation of a relation is a phrase and not a word, we would like to normalise the discriminating power of a relation with the length of its linguistic realisation. Calculating the length involves recursively generating referring expressions for the object of the preposition, an expensive task that we want to avoid unless we are actually using that relation in the final referring expression. We therefore initially approximate the length as follows. The realisation of a relation $[prep_o\ e_o]$ consists of $prep_o$, a determiner and the referring expression for e_o. If none of e_{ref}'s distractors have a $prep_o$ relation then we only require the head noun of e_o in the referring expression and $length = 3$. In this case, the relation is sufficient to identify both entities; for example, even if there were multiple bins in figure 1, as long as only one dog is in a bin, the reference *the dog in the bin* succeeds in uniquely referencing both the dog and the bin. If n distractors of e_{ref} contain a $prep_o$ relation with a non-e_o object that is distractor for e_o, we set $length = 3 + n$. This is an estimate for the word length of the realised relation that assumes one extra attribute for distinguishing e_o from each distractor. Normalisation by estimated length is vital; if e_o requires a long description, the relations's DQ should be small so that shorter possibilities are considered first in the incremental process. The formula for DQ for relations is therefore $DQ = (CQ - SQ)/length$.

This approach can also be extended to allow for relations such as comparatives which have syntactically optional arguments (e.g., *the earlier flight* vs *the flight earlier than UA941*) which are not allowed for by approaches which ignore realisation.

3.5 The Lexicalised Context-Sensitive IA

Our lexicalised context-sensitive incremental algorithm (below) generates a referring expression for *Entity*. As it recurses, it keeps track of entities it has used up in order to avoid entering loops like *the dog in the bin containing the dog in the bin....* To generate a referring expression for an entity, the algorithm calculates the DQs for all its attributes and approximates the DQs for all its relations (2). It then forms the **preferred** list (3) and constructs the referring expression by adding elements of **preferred** till the contrast set is empty (4). This is straightforward for attributes (5). For relations (6), it needs to recursively generate the prepositional phrase first.

It checks that it hasn't entered a loop (6a), generates a new contrast set for the object of the relation (6(a)i), recursively generates a referring expression for the object of the preposition (6(a)ii), recalculates DQ (6(a)iii) and either incorporates the relation in the referring expression or shifts the relation down the **preferred** list (6(a)iv). This step ensures that an initial mis-estimation in the word length of a relation doesn't force its inclusion at the expense of shorter possibilities. If after incorporating all attributes and relations, the contrast set is still non-empty, the algorithm returns the best expression it can find (7).

`set` *generate-ref-exp(Entity, ContrastSet, UsedEntities)*

1. `IF` *ContrastSet* = [] `THEN RETURN` {*Entity.head*}
2. Calculate CQ, SQ and DQ for each attribute and relation of *Entity* (as in Sec 3.1 and 3.4)
3. `Let` **preferred** be the list of attributes/ relations sorted in decreasing order of *DQ*s. `FOR` each element (*Mod*) of **preferred** `DO` steps 4, 5 and 6
4. `IF` *ContrastSet* = [] `THEN RETURN` *RefExp* ∪ {*Entity.head*}
5. `IF` Mod is an Attribute `THEN`
   (a) `LET` *RefExp* = {*Mod*} ∪ *RefExp*
   (b) Remove from *ContrastSet*, any entities *Mod* rules out
6. `IF` Mod is a Relation [$prep_i$ e_i] `THEN`
   (a) `IF` $e_i \in UsedEntities$ `THEN`
      i. Set $DQ = -\infty$
      ii. Move *Mod* to the end of **preferred**

      `ELSE`
      i. `LET` *ContrastSet2* be the set of non-e_i entities that are the objects of $prep_i$ relations in members of *ContrastSet*
      ii. `LET` *RE* = generate-referring-exp(e_i, *ContrastSet2*, {e_i}∪*UsedEntities*)
      iii. recalculate DQ using *length* = $2 + length(RE)$
      iv. `IF` position in **preferred** is lowered `THEN` re-sort **preferred**

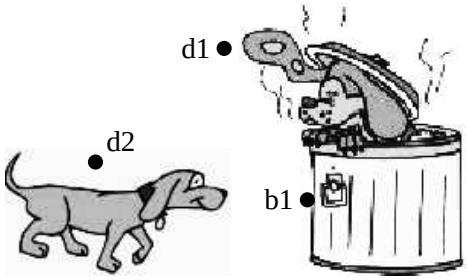
      `ELSE`
      (α) `SET` *RefExp* = *RefExp* ∪ $\{[prep_i | determiner | RE]\}$
      (β) Remove from *ContrastSet*, any entities that *Mod* rules out
7. `RETURN` *RefExp* ∪ {*Entity.head*}

An Example Trace:

We now trace the algorithm above as it generates a referring expression for `d1` in figure 1.

call generate-ref-exp(d1,[d2],[])

- step 1: *ContrastSet* is not empty
- step 2: $DQ_{small} = -4$, $DQ_{grey} = -4$
  $DQ_{[in\ b1]} = 4/3$, $DQ_{[near\ d2]} = 4/4$
- step 3: **preferred** = [[in `b1`], [near `d2`], small, grey]



$$d1\begin{bmatrix}\texttt{head} & dog\\ \texttt{attrib} & [small,\ grey]\\ \texttt{in} & b1\\ \texttt{near} & d2\end{bmatrix}\quad d2\begin{bmatrix}\texttt{head} & dog\\ \texttt{attrib} & [small,\ grey]\\ \texttt{outside} & b1\\ \texttt{near} & d1\end{bmatrix}$$

$$b1\begin{bmatrix}\texttt{head} & bin\\ \texttt{attrib} & [large, steel]\\ \texttt{containing} & d1\\ \texttt{near} & d2\end{bmatrix}$$

Figure 1: AVMs for two dogs and a bin

- Iteration 1 — *mod* = [in `b1`]
  - step 6(a)i: *ContrastSet2* = []
  - step 6(a)ii: **call generate-ref-exp(b1,[],[d1])**
    - step 1: *ContrastSet* = []
      return {**bin**}
  - step 6(a)iii: $DQ_{[in\ b1]} = 4/3$
  - step 6(a)ivα: *RefExp* = {[in, the, {bin}]}
  - step 6(a)ivβ: *ContrastSet* = []
- Iteration 2 — *mod* = [near `d2`]
  - step 4: *ContrastSet* = []
    return {**[in the** {**bin**}**], dog**}

The algorithm presented above is designed to return the shortest referring expression that uniquely identifies an entity. If the scene in figure 1 were cluttered with bins, the algorithm would still refer to `d1` as *the dog in the bin* as there is only one dog that is in a bin. The user gets no help in locating the bin. If helping the user locate entities is important to the discourse plan, we need to change step 6(a)(ELSE)i so that the contrast set includes all *bins* in context, not just *bins* that are objects of *in* relations of distractors of `d1`.

3.6 Compound Nominals

Our analysis so far has assumed that attributes are adjectives. However, many nominals introduced through relations can also be introduced in compound nominals, for example:

1. a church in Paris ↔ a Paris church
2. a novel by Archer ↔ an Archer novel
3. a company from London ↔ a London company

This is an important issue for regeneration applications, where the AVMs for entities are constructed from text rather than a semantic knowledge base (which could be constructed such that such cases are stored in relational form, though possibly with an underspecified relation). We need to augment our algorithm so that it can compare AVMs like:

$$\begin{bmatrix} \texttt{head} & \textit{church} \\ \texttt{in} & \begin{bmatrix} \texttt{head} & \textit{Paris} \end{bmatrix} \end{bmatrix} \text{ and } \begin{bmatrix} \texttt{head} & \textit{church} \\ \texttt{attrib} & [\textit{Paris}] \end{bmatrix}$$

Formally, the algorithm for calculating SQ and CQ for a nominal attribute a_{nom} of entity e_o is:

FOR each distractor e_i of e_o DO

1. IF a_{nom} is similar to any nominal attribute of e_i THEN $SQ = SQ + 4$
2. IF a_{nom} is similar to the head noun of the object of any relation of e_i THEN
   (a) $SQ = SQ + 4$
   (b) flatten that relation for e_i, i.e., add the attributes of the object of the relation to the attribute list for e_i

In step 2, we compare a nominal attribute a_{nom} of e_o to the head noun of the object of a relation of e_i. If they are similar, it is likely that any attributes of that object might help distinguish e_o from e_i. We then add those attributes to the attribute list of e_i. Now, if SQ is non-zero, the nominal attribute a_{nom} has bad discriminating power and we set $DQ = -SQ$. If $SQ = 0$, then a_{nom} has good discriminating power and we set $DQ = 4$.

We also extend the algorithm for calculating DQ for a relation $[\texttt{prep}_\texttt{j} \;\; e_j]$ of e_o as follows:

1. IF any distractor e_i has a nominal attribute a_{nom} THEN
   (a) IF a_{nom} is similar to the head of e_j THEN
      i. Add all attributes of e_o to the attribute list and calculate their DQs
2. calculate DQ for the relation as in section 3.4

We can demonstrate how this approach works using entities extracted from the following sentence (from the Wall Street Journal):

> Also contributing to the firmness in copper, the analyst noted, was **a report by Chicago purchasing agents**, which precedes **the full purchasing agents report** that is due out today and gives an indication of what *the full report* might hold.

Consider generating a referring expression for e_o when the distractor is e_1:

$$e_o = \begin{bmatrix} \texttt{head} & \textit{report} \\ \texttt{by} & \begin{bmatrix} \texttt{head} & \textit{agents} \\ \texttt{attrib} & [\textit{Chicago, purchasing}] \end{bmatrix} \end{bmatrix}$$

$$e_1 = \begin{bmatrix} \texttt{head} & \textit{report} \\ \texttt{attributes} & [\textit{full, purchasing, agents}] \end{bmatrix}$$

The distractor *the full purchasing agents report* contains the nominal attribute *agents*. To compare *report by Chicago purchasing agents* with *full purchasing agents report*, our algorithm flattens the former to *Chicago purchasing agents report*. Our algorithm now gives:

$$DQ_{agents} = -4,\ DQ_{purchasing} = -4,$$
$$DQ_{Chicago} = 4,\ DQ_{by\ Chicago\ purchasing\ agents} = 4/4$$

We thus generate the referring expression *the Chicago report*. This approach takes advantage of the flexibility of the relationships that can hold between nouns in a compound: although examples can be devised where removing a nominal causes ungrammaticality, it works well enough empirically.

To generate a referring expression for $e1$ (*full purchasing agents report*) when the distractor is e_o(*report by Chicago purchasing agents*), our algorithm again flattens e_o to obtain:

$$DQ_{agents} = -4,\ DQ_{purchasing} = -4$$
$$DQ_{full} = 4$$

The generated referring expression is *the full report*. This is identical to the referring expression used in the original text.

4 Evaluation

As our algorithm works in open domains, we were able to perform a corpus-based evaluation using the Penn WSJ Treebank (Marcus et al., 1993). Our evaluation aimed to reproduce existing referring expressions (NPs with a definite determiner) in the Penn Treebank by providing our algorithm as input:

1. The first mention NP for that reference.
2. The contrast set of distractor NPs

For each referring expression (NP with a definite determiner) in the Penn Treebank, we automatically identified its first mention and all its distractors in a four sentence window, as described in §4.1. We then used our program to generate a referring expression for the first mention NP, giving it a contrast-set containing the distractor NPs. Our evaluation compared this generated description with the original WSJ reference that we had started out with. Our algorithm was developed using toy examples and counter-examples constructed by hand, and the Penn Treebank was unseen data for this evaluation.

4.1 Identifying Antecedents and Distractors

For every definite noun phrase NP_o in the Penn Treebank, we shortlisted all the noun phrases NP_i in a discourse window of four sentences (the two

preceding sentences, current sentence and the following sentence) that had a head noun identical to or a WordNet synonym of the head noun of NP_o.

We compared the set of attributes and relations for each shortlisted NP_i that preceded NP_o in the discourse window with that of NP_o. If the attributes and relations set of NP_i was a superset of that of NP_o, we assumed that NP_o referred to NP_i and added NP_i to an antecedent set. We added all other NP_i to the contrast set of distractors.

Similarly, we excluded any noun phrase NP_i that appeared in the discourse after NP_o whose attributes and relations set was a subset of NP_o's and added the remaining NP_i to the contrast set. We then selected the longest noun phrase in the antecedent set to be the antecedent that we would try and generate a referring expression from.

The table below gives some examples of distractors that our program found using WordNet synonyms to compare head nouns:

Entity	Distractors
first half-free Soviet *vote*	fair *elections* in the GDR
military construction *bill*	fiscal *measure*
steep *fall* in currency	*drop* in market stock
permanent *insurance*	death benefit *coverage*

4.2 Results

There were 146 instances of definite descriptions in the WSJ where the following conditions (that ensure that the referring expression generation task is non-trivial) were satisfied:

1. The definite NP (referring expression) contained at least one attribute or relation.
2. An antecedent was found for the definite NP.
3. There was at least one distractor NP in the discourse window.

In 81.5% of these cases, our program returned a referring expression that was identical to the one used in the WSJ. This is a surprisingly high accuracy, considering that there is a fair amount of variability in the way human writers use referring expressions. For comparison, the baseline of reproducing the antecedent NP performed at 48%[2].

Some errors were due to non-recognition of multiword expessions in the antecedent (for example, our program generated *care product* from *personal care product*). In many of the remaining error cases, it was difficult to decide whether what our program generated was acceptable or wrong. For example, the WSJ contained the referring expression *the one-day limit*, where the automatically detected antecedent was *the maximum one-day limit for the S&P 500 stock-index futures contract* and the automatically detected contrast set was:

> {the five-point opening limit for the contract, the 12-point limit, the 30-point limit, the intermediate limit of 20 points}

Our program generated *the maximum limit*, where the WSJ writer preferred *the one-day limit*.

[2]We are only evaluating content selection (the nouns and pre- and post-modifiers) and ignore determiner choice.

5 Further Issues

5.1 Reference Modifying Attributes

The analysis thus far has assumed that all attributes modify the referent rather than the reference to the referent. However, for example, if `e1` is *an alleged murderer*, the attribute *alleged* modifies the reference *murderer* rather than the referent `e1` and referring to `e1` as *the murderer* would be factually incorrect. Logically `e1` could be represented as $(\text{alleged1}(\text{murderer1}))(x)$, rather than $\text{alleged1}(x) \wedge \text{murderer1}(x)$. This is no longer first-order, and presents new difficulties for the traditional formalisation of the reference generation problem. One (inelegant) solution would be to introduce a new predicate $\text{allegedMurderer1}(x)$.

A working approach in our framework would be to add a large positive weight to the DQs of reference modifying attributes, thus forcing them to be selected in the referring expression.

5.2 Discourse Context and Salience

The incremental algorithm assumes the availability of a contrast set and does not provide an algorithm for constructing and updating it. The contrast set, in general, needs to take context into account. Krahmer and Theune (2002) propose an extension to the IA which treats the context set as a combination of a discourse domain and a salience function. *The black dog* would then refer to the most salient entity in the discourse domain that is both *black* and a *dog*.

Incorporating salience into our algorithm is straightforward. As described earlier, we compute the quotients SQ and CQ for each attribute or relation by adding an amount $w \in \{4, 2, 1\}$ to the relevant quotient based on a comparison with the attributes and relations of each distractor. We can incorporate salience by weighting w with the salience of the distractor whose attribute or relation we are considering. This will result in attributes and relations with high discriminating power with regard to more salient distractors getting selected first in the incremental process.

5.3 Discourse Plans

In many situations, attributes and relations serve different discourse functions. For example, attributes might be used to help the hearer *identify* an entity

while relations might serve to help *locate* the entity. This needs to be taken into account when generating a referring expression. If we were generating instructions for using a machine, we might want to include both attributes and relations; so to instruct the user to switch on the power, we might say *switch on the red button on the top-left corner*. This would help the user locate the switch (*on the top-left corner*) and identify it (*red*). If we were helping a chef find the salt in a kitchen, we might want to use only relations because the chef knows what salt looks like. *The salt behind the corn flakes on the shelf above the fridge* is in this context preferable to *the white powder*. If the discourse plan that controls generation requires our algorithm to preferentially select relations or attributes, it can add a positive amount α to their DQs. Then, the resultant formula is $DQ = (CQ - SQ)/length + \alpha$, where $length = 1$ for attributes and by default $\alpha = 0$ for both relations and attributes.

6 Conclusions and Future Work

We have described an algorithm for generating referring expressions that can be used in any domain. Our algorithm selects attributes and relations that are distinctive in context. It does not rely on the availability of an adjective classification scheme and uses WordNet antonym and synonym lists instead. It is also, as far as we know, the first algorithm that allows for the incremental incorporation of relations and the first that handles nominals. In a novel evaluation, our algorithm successfully generates identical referring expressions to those in the Penn WSJ Treebank in over 80% of cases.

In future work, we plan to use this algorithm as part of a system for generation from a database of user opinions on products which has been automatically extracted from newsgroups and similar text. This is midway between regeneration and the classical task of generating from a knowledge base because, while the database itself provides structure, many of the field values are strings corresponding to phrases used in the original text. Thus, our lexicalised approach is directly applicable to this task.

7 Acknowledgements

Thanks are due to Kees van Deemter and three anonymous ACL reviewers for useful feedback on prior versions of this paper.

This document was generated partly in the context of the Deep Thought project, funded under the Thematic Programme User-friendly Information Society of the 5th Framework Programme of the European Community (Contract N IST-2001-37836)

References

Robert Dale and Nicholas Haddock. 1991. Generating referring expressions involving relations. In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics (EACL'91)*, pages 161–166, Berlin, Germany.

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

Helmut Horacek. 2003. A best-first search algorithm for generating referring expressions. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, pages 103–106, Budapest, Hungary.

Emiel Krahmer and Mariët Theune. 2002. Efficient context-sensitive generation of referring expressions. In Kees van Deemter and Rodger Kibble, editors, *Information Sharing: Givenness and Newness in Language Processing*, pages 223–264. CSLI Publications, Stanford,California.

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

Mitchell Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large natural language corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

George A. Miller, Richard Beckwith, Christiane D. Fellbaum, Derek Gross, and Katherine Miller. 1993. Five Papers on WordNet. Technical report, Princeton University, Princeton, N.J.

Ehud Reiter. 1990. The computational complexity of avoiding conversational implicatures. In *Proceedings of the 28th Annual Meeting of Association for Computational Linguistics (ACL'90)*, pages 97–104, Pittsburgh, Pennsylvania.

Ehud Reiter and Robert Dale. 1992. A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING'92)*, pages 232–238, Nantes, France.

Kees van Deemter. 2000. Generating vague descriptions. In *Proceedings of the 1st International Conference on Natural Language Generation (INLG'00)*, pages 179–185, Mitzpe Ramon, Israel.

Kees van Deemter. 2002. Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1):37–52.