

Generating Anaphora for Simplifying Text

In Proceedings of the 4th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2002). Pages 199-204

Advaith Siddharthan & Ann Copestake

Natural Language and Information Processing Group
Computer Laboratory, University of Cambridge
{as372, aac10}@cl.cam.ac.uk

Abstract

We present an algorithm for generating referring expressions in open domains. Existing algorithms assume a classification of adjectives which is possible only for restricted domains. Our alternative relies on WordNet synonym and antonym sets and gives equivalent results on the examples cited in the literature and improved results in other cases that prior approaches cannot handle. We believe that it is also the first algorithm that allows for the incremental incorporation of relations. We perform an evaluation on a text-simplification task on Wall Street Journal data.

1. Introduction

The automatic dis-embedding of relative clauses is an important aspect of text simplification, an NLP task that aims to rewrite sentences, reducing their complexity while preserving their meaning and information content (Chandrasekar et al., 1996; Carroll et al., 1998). Text simplification is a useful NLP task for varied reasons. Chandrasekar et al. (1996) and Chandrasekar and Srinivas (1997) viewed text simplification as a preprocessing tool to improve the performance of their parser. The PSET project (Carroll et al., 1998; Carroll et al., 1999), on the other hand, focused its research on simplifying newspaper text for aphasics, who have trouble with long sentences, infrequent words and complicated grammatical constructs including embedded clauses (Devlin, 1999). Consider:

A former ceremonial officer from Derby, who was at the heart of Whitehall's patronage machinery, says there is a general review of the state of the honours list every five years or so.

This simplifies to (see Devlin (1999) for motivation):

A former ceremonial officer from Derby was at the heart of Whitehall's patronage machinery. This former officer says there is a general review of the state of the honours list every five years or so.

We require a referring expression for the noun phrase to which the clause attaches to use as the subject when we dis-embed the clause. In the above example, we need to generate *This former officer* from *A former ceremonial officer from Derby*. Reproducing the entire NP can make the text look stilted. Moreover, including too much information in the referring expression can convey unwanted and possibly wrong conversational implicatures. This is a problem that arises when simplifying other grammatical constructs as well; e.g., separating out conjoined verb phrases or making new sentences out of appositives.

In this paper, we present an algorithm for generating referring expressions in open domains. We present it as a general purpose algorithm, though we evaluate it on the text simplification task.

2. Generating Referring Expressions

We present our attribute selection algorithm in section 2.1. and extend it to handle relational descriptions in section 2.3. and nominal attributes in section 2.5.. We discuss the issue of forming the contrast set in section 2.2..

2.1. Attributes

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 the example in their paper (that deals with entities like *the small black dog, the white cat...*), the preference might be [colour, size, shape, ...]. The algorithm then keeps adding attributes from **preferred-attributes** that rule out at least one entity in the contrast set to the referring set until all the entities in the contrast set 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.

$\left[\begin{array}{ll} \text{type} & \text{dog} \\ \text{size} & \text{large} \\ \text{colour} & \text{brown} \end{array} \right]$	$\left[\begin{array}{ll} \text{type} & \text{dog} \\ \text{size} & \text{large} \\ \text{colour} & \text{black} \end{array} \right]$
---	---

Assuming that the **preferred-attributes** list is [size, colour, ...], the algorithm would first compare the values of the *size* attribute (both *large*), disregard that attribute as not being discriminating, compare the values of the *colour* attribute and return *the brown dog*.

Unfortunately, the incremental algorithm is unsuitable for open domains because it assumes the following:

1. A classification scheme for attributes exists
2. The values that attributes take are mutually exclusive
3. Linguistic realisations of attributes are unambiguous

All these assumptions are violated when we move from generation in a very restricted domain to generation or re-generation in an open domain. Adjective classification is a hard problem and there is no sensible classification scheme that can be used if we are dealing with an open domain like newspaper text. Even if we had such a scheme, we would not be able to assume the mutual exclusivity of values; for example, we might end up comparing [size *big*] with [size *large*] or [colour *dark*] with [colour *black*]. Selecting attributes at the semantic level is risky because their linguistic realisation might be ambiguous and most of the common adjectives are polysemous (See example 1 in section 2.1.1.).

Our alternative algorithm measures the relatedness of adjectives, rather than decides if two of them are the same or not (section 2.1.1.). It works at the level of words, not their semantic labels. Further, it treats discriminating power as only one criteria for selecting attributes and allows for the easy incorporation of other considerations like reference modification (section 2.1.3.) and the reader’s comprehension skills (section 2.1.4.).

2.1.1. Quantifying Discriminating Power

We define the following three quotients.

Similarity Quotient (SQ)

We define *similarity* as transitive synonymy. The idea is that a synonym of a synonym is a synonym, and the level of synonymy between two adjectives depends on how many times we have to chain through WordNet (Miller et al., 1993) 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* as transitive antonymy. We form the set C_1 of strict WordNet antonyms of a_j , C_2 of strict WordNet antonyms of members of S_1 and WordNet Synonyms of members of C_1 and C_3 similarly 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 DQ s) in which to incorporate attributes. We demonstrate our approach with two examples.

Example 1

Suppose we need to differentiate between:

$$e1 \begin{bmatrix} \text{type} & \textit{president} \\ \text{age} & \textit{old} \\ \text{tenure} & \textit{present} \end{bmatrix} \& e2 \begin{bmatrix} \text{type} & \textit{president} \\ \text{age} & \textit{young} \\ \text{tenure} & \textit{past} \end{bmatrix}$$

If we followed the strict typing system used by previous algorithms, 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 successfully picks *the present president* as *present* has a higher DQ than *old*.

attribute	distractor	CQ	SQ	DQ
old	$e2\{\textit{young, past}\}$	4	4	0
present	$e2\{\textit{young, past}\}$	4	0	4

Example 2

Assume we have four dogs in context: $e1$ (a large brown dog), $e2$ (a small black dog), $e3$ (a tiny white dog) and $e4$ (a big dark dog). To refer to $e4$, for each of its attributes, we calculate the three quotients with respect to $e1, e2$ and $e3$.

attribute	distractor	CQ	SQ	DQ
big	$e1\{\textit{large, brown}\}$	0	4	-4
big	$e2\{\textit{small, black}\}$	4	0	4
big	$e3\{\textit{tiny, white}\}$	1	0	1
dark	$e1\{\textit{large, brown}\}$	0	0	0
dark	$e2\{\textit{small, black}\}$	1	4	-3
dark	$e3\{\textit{tiny, white}\}$	2	1	1

Overall, *big* has a higher discriminating power (1) than *dark* (-2). We therefore pick *big* and rule out all the distractors that *big* has a positive DQ for (in this case, $e2$ and $e3$). $e1$ is the only distractor left. And we need to pick *dark* because *big* has a negative DQ for $e1$ and *dark* doesn’t.

If we had to refer to $e3$, we would end up with simply *the white dog* as *white* has a higher overall DQ (6) than *tiny* (1) and rules out $e2$ and $e4$. As *white*’s DQ with the only remaining distractor $e1$ is non-negative, we can assume it to be sufficiently discriminating.

attribute	distractor	CQ	SQ	DQ
tiny	$e1\{\textit{large, brown}\}$	1	0	1
tiny	$e2\{\textit{small, black}\}$	0	1	-1
tiny	$e4\{\textit{big, dark}\}$	1	0	1
white	$e1\{\textit{large, brown}\}$	0	0	0
white	$e2\{\textit{small, black}\}$	4	0	4
white	$e4\{\textit{big, dark}\}$	2	0	2

2.1.2. Justifying our Algorithm

The psycholinguistic justification for the incremental algorithm hinges on two premises—(1) humans build up referring expressions incrementally and (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. 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 incremental algorithm.

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

Both the incremental algorithm 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 incremental algorithm, that compares each attribute of the discourse referent to only one attribute per distractor, is linear in n .

A major advantage of our approach is that it is easy to incorporate considerations other than discriminating power into the attribute selection process.

2.1.3. 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 e_1 is *an alleged murderer*, the attribute *alleged* modifies the reference *murderer* rather than the referent e_1 and referring to e_1 as *the murderer* would be factually incorrect. We can handle reference modifying adjectives trivially by adding a large positive weight to their DQ s. This will have the effect of forcing them to be selected in the referring expression.

2.1.4. Reader’s Comprehension Skills

We can specify a user-dependent DQ cut-off for inclusion of adjectives. For example, for very low reading age readers, we could include every adjective with a non-negative DQ . Increasing the cut-off would result in fewer adjectives being included. Alternatively, we could weigh DQ s according to how common the adjective is. This can be measured using frequency counts on a corpus. The main intuition we use is that uncommon adjectives have more discriminating power than common adjectives. However, they are also more likely to be incomprehensible to people with low reading ages. If we give uncommon adjectives higher weights, we will end up with referring expressions containing fewer, though harder to understand, adjectives. This is ideal for readers with high reading ages. On the other hand, if we flip the weights, so that common adjectives get higher weights, we will end up with referring expressions containing many simple adjectives. This is ideal for people with low reading ages.

2.2. Forming the Contrast Set

The incremental algorithm assumes the availability of a contrast set of distractors 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 incremental algorithm which treats the context set as a combination of a discourse domain and a salience function. Incorporating salience into our algorithm is trivial. In section 2.1.1., we described how to compute the quotients SQ and CQ for an attribute. This was done by adding an amount $w \in \{4, 2, 1\}$ to the relevant quotient each time a distractor’s attribute was discovered in a synonym or antonym list. We can incorporate salience by weighting w with the salience of the distractor whose attribute we are considering. This will result in attributes with high discriminating power with regard to more salient distractors getting selected first in the incremental process. However, for the evaluation in section 3., we do not consider salience. This is because our input is newspaper articles and we have found empirically that there are rarely more than three distractors.

To form the contrast set for NP_o , we identify all the NPs in a discourse window of four sentences and select potential distractors among them. We consider two cases separately.

If NP_o is indefinite, it is being newly introduced into the discourse and any noun phrase previously mentioned within the discourse window that has a similar lexical head (determined by WordNet synonym sets) is a distractor.

If NP_o is definite, we want to exclude from our contrast set any noun phrases that it co-refers with. If the attribute set of a previously mentioned noun phrase with similar lexical head (NP_i) is a superset of the attribute set of NP_o , we assume that NP_o refers to NP_i and exclude NP_i from the contrast set. Any other noun phrase previously mentioned within the discourse window that has a similar lexical head is a distractor.

Irrespective of whether NP_o is definite or indefinite, we exclude any noun phrase NP_j that appears in the window after NP_o whose attribute set is a subset of NP_o ’s.

The table below gives some examples of distractors that our program found during the evaluation (section 3.).

Entity	Distractors
first half-free Soviet vote	fair <i>elections</i> in the GDR
military construction bill	fiscal <i>measure</i>
copper consumption	declining <i>use</i>
cunning ploy	public education <i>gambit</i>
steep fall in currency	<i>drop</i> in market stock
permanent insurance	death benefit <i>coverage</i>

2.3. Relations

Semantically, *attributes* describe an entity (eg. *the small grey dog*) and *relations* relate an entity to other entities (eg. *the dog in the big 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 incremental algorithm does not consider relations and the referring expression is constructed out of only attributes. It is difficult to imagine how relational descriptions can be incorporated in the incremental framework of the Reiter and Dale (1992) algorithm, where the order of incorporation of modifiers is predetermined according to a classification system. The Dale and Haddock (1991) algorithm allows for relational

descriptions but involves exponential global search. An important difference between the Reiter and Dale (1992) incremental algorithm and our incremental approach is that our approach computes the order in which attributes are incorporated on the fly, by quantifying their utility through 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.

2.3.1. Calculating DQ for Relations

Suppose we need to compute the three quotients for the relation $[prep_o e_o]$. We consider each entity e_i in the contrast set 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_o 's distractors have a $prep_o$ relation then we only require the head noun of the object in the referring expression and $length = 3$. If n distractors contain a $prep_o$ relation with a non- e_o object, we set $length = 3 + n$. This is an approximation to the length of the realisation of the relation that assumes one extra word per distractor. We now define $DQ = (CQ - SQ)/length$.

Attributes are usually used to *identify* an entity while relations, in most cases, serve to *locate* an entity. This needs to be taken into account when generating a referring expression. For example, 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 (*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 fine white crystals*. If the discourse plan requires the algorithm to preferentially select relations or attributes, we can add a positive amount α to their DQ s. So the final formula is $DQ = (CQ - SQ)/length + \alpha$, where $length = 1$ for attributes and by default $\alpha = 0$ for both relations and attributes.

2.4. The Complete Algorithm

The 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 DQ s for all its attributes and approximates the DQ s 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). 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 2.1.1. and 2.3.1.)
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* \cup {*Entity.head*}
5. IF *Mod* is an Attribute THEN
 - (a) LET *RefExp* = {*Mod*} \cup *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 the **preferred** list
 - 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 } \cup *UsedEntities*)
 - iii. recalculate DQ using $length = 2 + length(RE)$
 - iv. IF position in **preferred** is lowered THEN resort **preferred**
 - ELSE
 - (α) SET *RefExp* = *RefExp* \cup {[*prep_i*|*determiner*|*RE*]}
 - (β) Remove from *ContrastSet*, any entities that *Mod* rules out
7. RETURN *RefExp* \cup {*Entity.head*}

An Example

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: $DQ_{small} = -4$, $DQ_{grey} = -4$
 $DQ_{[in\ b1]} = 4/3$, $DQ_{[near\ d2]} = 4/4$

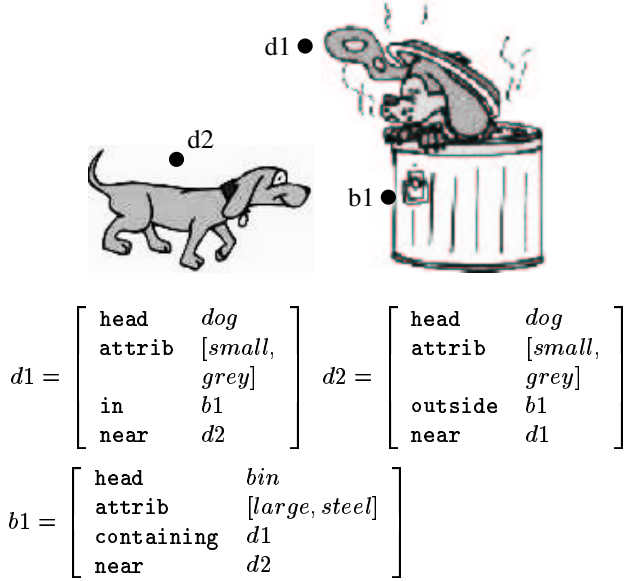


Figure 1: AVM representation of two dogs and a bin

- step 3: **preferred** = [[in b1], [near d2], small, grey]
- 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)iv α : *RefExp* = {[in, the, {bin}]}]
 - step 6(a)iv β : *ContrastSet* = []
- Iteration 2 — *mod* = [near d2]
 - step 4: *ContrastSet* = []
 - return {[in the {bin}], dog}**

The algorithm presented above tries 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)i so that the contrast set includes all *bins* in context, not just *bins* that are objects of *in* relations of distractors of d1.

2.5. Handling Nominals

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

1. the centre of the city \leftrightarrow the city centre
2. the president of IBM \leftrightarrow the IBM president
3. a company from East London \leftrightarrow an East London company

This means that we need to compare nominal attributes with the objects of relations and vice versa. 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

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 $DQ = 4$.

In short, we first 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 .

We also need to extend the algorithm for calculating DQ for a relation [prep $_j$ 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 DQ s
2. calculate DQ for the relation as in section 2.3.1.

In short, we first compare the head noun of e_j in a relation [prep $_j$ e_j] of e_o to a nominal attribute a_{nom} of e_i . If they are similar, it is likely that any attributes of e_j might help distinguish e_o from e_i . We then add those attributes to the attribute list of e_o . We demonstrate how this kind of abduction works with an example. Consider simplifying the following sentence:

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.

There are two clauses that can be dis-embedded, shown above in italics and bold font respectively. To dis-embed the italicised *which* clause, we need to generate a referring expression for e_o when the distractor is e_1 :

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

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

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

up with the referring expression *the Chicago report*. The simplified text is now:

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

For the *that* clause, we need to find a referring expression for e_1 (*full purchasing agents report*) when the distractor is e_o (*report by Chicago purchasing agents*). Our algorithm again flattens e_o and gives $DQ_{agents} = -4$, $DQ_{purchasing} = -4$ and $DQ_{full} = 4$. The final dis-embedded text is now:

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

3. Evaluation

We evaluated our algorithm on the Penn Wall Street Journal Corpus (Marcus et al., 1993). We looked at 135 *who*, *which* or *that* relative clauses where the referent NP (for which we had to generate a referring expression) had at least two attributes or relations and where our algorithm found at least one distractor. The performance of our algorithm is shown below:

Correct		Acceptable		Wrong	
freq	%	freq	%	freq	%
109	80.7	17	12.6	9	6.7

We labelled an example as *correct* if our program generated a referring expression that was optimal and factually accurate, as *acceptable* if the generated expression was accurate but not optimal and as *wrong* if the generated expression was nonsensical or ambiguous with a distractor. Around half the *wrong* results arose because a multiword expression was incorrectly treated as multiple attributes:

Noun Phrase	Generate Ref. Exp.
personal care products	care products
open end mutual funds	end funds
privately funded research	funded research
Wall Street Firms	Street Firms

The rest arose due to reference modifying attributes (eg: *China's current account* got referred to as *China's account*) or due to WordNet not finding a similarity relation between attributes like *nuclear* and *power* (eg: referring to *the company's AP600 nuclear power plant* as *the power plant* when one of the distractors was *domestic nuclear plant*).

4. 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 incorporations of relations and the first that handles nominals. Our approach gives encouraging results on the application that we have tested it on.

Future work includes analysing the discourse aspects of text simplification. Issues include determiner choice, ordering the simplified sentences and ensuring that the relative salience of different entities is not altered by the simplification process.

5. References

- John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of English newspaper text to assist aphasic readers. In *Proceedings of AAAI98 Workshop on Integrating Artificial Intelligence and Assistive Technology, Madison, Wisconsin*.
- John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying English text for language impaired readers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Bergen, Norway*.
- Raman Chandrasekar and Bangalore Srinivas. 1997. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10:183–190.
- Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING '96), Copenhagen, Denmark*.
- Robert Dale and N. Haddock. 1991. Generating referring expressions involving relations. In *Proceedings of EACL-91, Berlin*, pages 161–166.
- Siobhan Devlin. 1999. Simplifying natural language for aphasic readers. Technical report, Ph.D. thesis, University of Sunderland, UK.
- Emiel Krahmer and Mariët Theune. 2002. Efficient context-sensitive generation of referring expressions. In K. van Deemter and R. Kibble, editors, *Information Sharing: Givenness and Newness in Language Processing*. CSLI Publications.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann 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 and Robert Dale. 1992. A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th International Conference on Computational Linguistics, Nantes, France*, pages 232–238.
- Ehud Reiter. 1990. The computational complexity of avoiding conversational implicatures. In *Proceedings of the 28th Annual Meeting of Association for Computational Linguistics*, pages 97–104.