Search in Constraint-Based Paraphrasing

Mark Dras
Microsoft Research Institute, Macquarie University
Sydney NSW Australia 2109
markd@mpce.mq.edu.au

Abstract

Existing paraphrase systems either create successive drafts from an underlying representation, or they aim to correct texts. A different kind of paraphrase, also reflecting a real-world task, involves imposing, on an original text, external constraints in terms of length, readability, etc. This sort of paraphrase requires a new framework which allows production of a new text satisfying the constraints, but with minimal change from the original. In order for such a system to be feasible when used on a large scale, searching the space of candidate solution texts has to be made tractable. This paper examines a method for pruning the search space via branch-and-bound, evaluates three variants to find the most efficient model, and discusses its relation to standard heuristic methods such as genetic algorithms.

1 Introduction

The paraphrasing framework of this paper is one where a text is modified 'reluctantly' to conform to external surface constraints, such as length or readability requirements (see, for example, Dras, 1997a). This reluctant paraphrase (RP) paradigm can best be defined by contrasting it with the remedial sort of paraphrases suggested by style checkers or in style guides. The starting point under this remedial style of paraphrase is an imperfect text which has to be corrected, the corrections being determined by, for example, advice to make the text "more active"; examples of systems that implement this notion of paraphrase are the CCS system of Kieras (1990) used by the US military, or the grammar checkers found in word processors. In contrast, imagine the completion of an ideal document which says exactly what the author intends, but which must be changed to satisfy external surface constraints. These constraints might be a requirement to cut down an academic paper by one page for conference publication; or to make a technical document conform to house style readability requirements; or some combination of these or other sorts of external constraints. Thus, the text has to be paraphrased, albeit reluctantly, in order to meet these requirements, and this paper describes the beginnings of a new model required to handle this sort of paraphrase.

There are existing Natural Language Processing systems which already deal with surface constraints on text. Most of these are natural language generation (NLG) systems, and fall within the field of revision-based generation: WEIVER (Inui *et al*, 1992) and STREAK (Robin, 1994) are two such systems. In these systems, draft texts are produced, criticised, and then paraphrased[1] to produce a text that fits certain constraints. However, in these systems the constraints are local to a small unit of text, generally a sentence. For example, STREAK has a constraint that prevents sentences from being longer than 46 words, whereas in the model of this paper the constraints are applied to a whole multi-sentence text. One approach taken by these Natural Language Generation systems is to revise the text a step at a time, adding new information until just before the constraints would be violated (as STREAK does); in STREAK's approach, once

[1]The concept of paraphrase in these systems is somewhat different from the one used in this paper: once a draft is critiqued, a 'paraphrase' (textual variant) is re-generated from some underlying representation. In this paper a paraphrase is a direct text-to-text syntactic mapping.

a choice to add information has been made, it cannot be revoked. Another approach is to see if the text has already violated constraints, and if so to rewrite it (as WEIVER does); a history of rewritings is kept so that there will not be an infinite series of paraphrasings. These approaches work well for small, sentence-scale text, but become problematic when the search space is larger than for sentences. For instance, under a greedy locally-optimal choosing scheme such as that of STREAK, it is possible that a globally optimal choice of additions could be missed, or that in fact no solution could be found.[2] The alternative of WEIVER, keeping a history, is equivalent to explicitly enumerating all paraphrase alternatives; when the history gets large (again, not a problem for small, sentence-scale text) this leads to a very large search space with the simple backtrack search of WEIVER.

Applying global constraints to an entire text requires a different model, one which allows an efficient search of the space of solutions, given that it is much larger. Mathematical optimisation techniques have been used for the purpose of reducing search space sizes in areas such as scheduling and planning, and also, in Natural Language Processing, in semantic parsing (Beale, 1996). This paper takes a basic model of RP under an optimisation framework (Dras, 1997a) and develops it further. The model has three components: a set of paraphrases which is used to achieve the overall text modification; a set of constraints to which the text must adhere after the modification; and an effect—that of the change to the text caused by the paraphrases applied—which is to be minimised. This paper looks at a method for producing a more efficient formulation of the basic model. The following sections describe the components of the model—the constraints, the paraphrases, and the global effects they have on a text; a basic mathematical model for describing the interaction of these; two more refined models; experimental work comparing the three models; and a discussion of how these models could interact with standard heuristic techniques like genetic algorithms.

2 Components of the Model

2.1 Textual Constraints

This section outlines three measures of text, those of length, readability and lexical density. These measures are often used in the production of text; their numeric quality makes them particularly amenable to the optimisation models of this paper. See also Dras (1997a).

Length is the simplest measure, and is frequently used in practice as a constraint. For example, imposing a word limit on a text is standard for academic conferences, and meeting this constraint often involves cutting down a longer draft version. Constraining text length is also a feature of computational language generation systems, for instance as an explicit limit on the length of an individual text unit, as in the STREAK system (Robin, 1994).

Another common measure comes from readability formulae, such as the Flesch Reading Ease Score (Klare, 1974-5). Standard readability formulae are equations which attempt to predict, rather than evaluate, the readability of text; in form they are generally linear combinations of factors which correlate with text complexity, such as average sentence length. Although readability formulae are controversial their use here as a constraint can be defended on practical grounds: readability formulae are used as criteria for writing public documents in the US, such as insurance policies and other contracts; for producing military or software documentation; and so on. In these situations the use of readability formulae is mandatory; so for a system which models actual constraints on text, using the formulae as a constraint is reasonable.

Lexical density is a textual measure that attempts to capture the 'condensedness' of text by measuring the proportion of non-content (or function) words to total text. This idea of density has been used to distinguish between writ-

[2] In STREAK, given that information-adding is optional, this is not too much of a problem; but, when modifying an entire text, if the constraints compel some change to the text (like shortening the total length), STREAK'S greedy heuristic would commit to a choice before globally evaluating combinations of alternatives, which can be problematic.

ten and spoken forms of language: written language tends to be more dense than spoken. The concept is also useful in the context of this paper's optimisation model to prevent excessive text compression, and as a constraint counterbalancing the readability one. Under a typical readability formula, the formula value can be improved by the sort of paraphrases which compress text; a lexical density constraint can prevent too much of this.

2.2 Paraphrases

Within RP, individual paraphrases are just broad-coverage tools, used to adjust a text, rather than the corrective devices of remedial paraphrases. Hence the most appropriate paraphrases for this work are ones that are syntactic in nature, as they apply to a wide range of texts; see Dras (1997a) for more detail about the sources and subtypes of paraphrases. An example of this type is the splitting off of a noun postmodifier to form a separate sentence in the mapping from *Sarah eyed the page filled with bizarre linguistic phenomena.* to *Sarah eyed the page. It was filled with bizarre linguistic phenomena.*

These paraphrases will cause some change to the text in terms of the constraints described in Section 2.1. These changes are fairly straightforward—changes to the number of words in the text, the average number of words per sentence, and so on; the exact way in which this occurs is described in Section 3.1.

Under RP, any change effected by a paraphrase is taken to be a negative (that is, undesirable) one, moving the text away from the author's intended meaning. Developing an optimisation model thus requires a quantification of the effects that imposing a paraphrase on a text will have on that text. The definition of such an objective function minimising the effect of paraphrases, in terms of truth-conditional meaning and information structure, is also described in Dras (1997a). For a computational paraphrase system a formal specification of the paraphrases is also needed; this is done within the representation formalism based on Tree Adjoining Grammars (Joshi *et al*, 1975) described in Dras (1997b); however, an informal description is adequate here for discussion of the paraphrase effects and their inclusion into the optimisation model.

3 An Optimisation Approach

This section presents a mathematical optimisation model of paraphrasing. The basic techniques are those of integer programming (see, e.g., Nemhauser and Wolsey, 1988), which describes the constraints and function to be minimised in terms of linear combinations of integer variables. The integer programming approach is useful because it provides a set of techniques for finding an optimal solution, and heuristics for pruning the search space. After a formal presentation of a basic model, an example is given for clarification.

3.1 A Basic Model

In developing an optimisation model, it is first necessary to identify the DECISION VARIABLES: that is, those factors about which a decision is to be made. In this case, it is the paraphrase mappings: for each paraphrase, the decision is whether this paraphrase should be applied to the text to move it towards satisfying the constraints while minimally perturbing the text. In this situation, the choice is binary, whether or not to apply the paraphrase. Given this, the decision variables are

p_{ij} = 0/1 variable representing the jth paraphrase for sentence i

The OBJECTIVE FUNCTION, the function to be optimised, is, for RP, a measure of the change to the text. With c_{ij} being the effect (or cost) of each paraphrase, if applied, this function has the form

$$z = \sum c_{ij} . p_{ij}$$

Expressed mathematically, the length constraint is

$$\sum w_{ij} . p_{ij} \leq k_1$$

where w_{ij} is change to length of sentence i caused by paraphrase ij and k_1 is required change to the length of text in words ($k_1 \leq 0$).

A simplified readability constraint using only the average sentence length component, is

$$\frac{W + \sum w_{ij}.p_{ij}}{S + \sum s_{ij}.p_{ij}} \leq k_2$$

That is,

$$\sum (w_{ij} - k_2.s_{ij})p_{ij} \leq k_2 S - W$$

where s_{ij} is change to number of sentences in the text by paraphrase ij, W is total words in original text, S is total sentences in original text, and k_2 is required average sentence length ($k_2 \geq 0$).

The lexical density constraint requires the proportion of function words (taken here to be all closed class words) to total words to be greater than some constant value. Analogously to the readability constraint, this is

$$\sum (f_{ij} - k_3.w_{ij})p_{ij} \geq k_3 W - F$$

where f_{ij} is change to number of function words caused by paraphrase ij, F is total number of function words in original text, and k_3 is required proportion of function words to total words ($0 \leq k_3 \leq 1$).

Given that there are n_i paraphrases for sentence i, there is a potential conflict for the paraphrases. To simplify their application, an extra constraint is added, stating that there can be at most one paraphrase for each sentence:

$$\sum_j p_{ij} \leq 1$$

An example is presented in the next section, to illustrate the model. The small size of this example does not allow a real demonstration of the usefulness of the approach, since the problem can be solved almost by inspection. However, in larger problems this method of modelling allows the use of techniques such as branch-and-bound (Nemhauser and Wolsey, 1988) which make the solution of the problem feasible, where the solution would otherwise be impractical because of the problem's exponential complexity.

3.2 An Example

As an example, take the short text:

> The cat sat on the mat which was by the door. It ate the cream ladled out by its owner. The owner, an eminent engineer, had a convertible used in a bank robbery.

$$\begin{array}{l}
\text{minimise} \\
z \;=\; c_{11}p_{11} + c_{21}p_{21} + c_{31}p_{31} + c_{32}p_{32} \\
\text{given} \\
\begin{array}{rrrrcr}
-\ 2p_{11} & +\ 3p_{21} & +\ 3p_{31} & -\ 3p_{32} & \leq & 0 \\
-\ 2p_{11} & -\ 7p_{21} & -\ 7p_{31} & -\ 3p_{32} & \leq & -3 \\
-\ 0.95p_{11} & +\ 1.43p_{21} & +\ 1.43p_{31} & +\ 0.58p_{32} & \geq & 0.33 \\
 & & p_{31} & +\ p_{32} & \leq & 1
\end{array}
\end{array}$$

Figure 1: Pure binary variable formulation

The values of F, W and S are 17, 33 and 3 respectively.

Possible paraphrases of individual sentences, using just relative pronoun deletion, postmodifier split, and parenthetical deletion, are:

(1) p_{11}. The cat sat on the mat by the door.
p_{21}. It ate the cream. It had been ladled out by its owner.
p_{31}. The owner, an eminent engineer, had a convertible. It had been used in a bank robbery.
p_{32}. The owner had a convertible used in a bank robbery.

This gives decision variables and associated coefficients as in Table 1. For the example, the constraint values are (arbitrarily) chosen as $k_1 = 0$ (at worst no compression of text length), $k_2 = 10$ (average sentence length no greater than 10), and $k_3 = 0.525$ (function words no less than 52.5% of the text). The basic integer programming (IP) formulation is then given in Figure 1.

There are two alternatives which are feasible solutions, with either three variables zero and $p_{32} = 1$, or three variables one and $p_{31} = 0$. This gives two values for the objective function, $z = c_{32}$ and $z = c_{11} + c_{21} + c_{32}$. Since under the RP assumption all changes involve a positive cost, the best alternative is the first, with only the second paraphrase for sentence number three being applied. The resulting text is then as the original with *an eminent engineer* deleted.

4 Refining the Basic Model

It is well-known (Williams, 1995) that IP problems have many different formulations, much

paraphrase ij	f_{ij}	w_{ij}	s_{ij}
11	-2	-2	0
21	+3	+3	+1
31	+3	+3	+1
32	-1	-3	0

Table 1: Variable coefficients

more so than similar linear programming problems. A consequence of this is that there is greater scope for improving the formulation of IPs so that the search space becomes more tractable.

The basic model above, in the course of finding a solution, has a larger search space than is really necessary because of the binary nature of the variables, even in cases where they are effectively interchangeable. For the example in Figure 1, p_{21} and p_{31} are interchangeable at the level of numeric abstraction used in the model, in the sense that the effect of either on the text in terms of the objective function and all the constraints will be the same. These two variables can thus be aggregated into a single integer variable with range [0,2].

The binary-only formulation will in general have a larger search space: part of the search is devoted to finding equivalent optimal solutions (in the example, $p_{21} = 0$ and $p_{31} = 1$ as against $p_{21} = 1$ and $p_{31} = 0$). In effect, the additional search is used to find the ordering of solutions.

This kind of symmetry of solutions is undesirable, and there has been much recent work on symmetry breaking in constraint satisfaction programming (see, for example, Joslin and Roy, 1997), in order to produce formulations with smaller search spaces. Such work has been in problems such as scheduling and planning; the rest of this paper looks at taking advantage of features of textual paraphrases to produce better formulations of RP models.

4.1 Aggregating variables

At any particular point at which a paraphrase is possible in text, there are often a number of variants of one particular paraphrase type that can be applied. One type of variance for a particular paraphrase is in the choice of referring expression. So, for example, for the type of paraphrase that involves splitting off a noun post-modifier and forming a separate sentence (as in (2)), there are a number of paraphrase variants depending on the chosen referring expression.

(2) a. He desperately wanted the brown corduroy jacket worn by the model.
b. He desperately wanted the brown corduroy jacket. X was worn by the model.

In this paraphrase (2), for example, there are a number of alternatives for X: the pronoun *it*, or some more specific NP (*the jacket*, *the brown jacket*, *the corduroy jacket*, *the brown corduroy jacket*)[3]. This leads to, for (2), four variants of the post-modifying paraphrase, all with similar properties in terms of constraint equations and objective function. This can be viewed as a single variable with coefficients that can also be varied to a limited extent.

Let a variable's C-SET be defined as the set of its coefficients across all constraints. The c-set for the paraphrase of (2), using the equations of Section 3.1, is then $\{2+Y, 2+Y-k_2, 1-(2+Y)k_3\}$, with $0 \leq Y \leq 3$. From the discussion in the previous section, it would seem desirable to use the variants which allow aggregation of variables; this should produce a smaller total search space. However, even though the total space of solutions will be smaller when the aggregable variant is chosen—for the example, 12 solutions as against 16—it may not be the case that the space is smaller when search pruning techniques, such as branch-and-bound, are used. Additionally, the search size may depend on the way in which the aggregation is carried out. The following sections consider an experiment to test the idea that aggregable variants do have smaller search spaces; investigate two algorithms for aggregating variables; and then discuss the results of the experiment.

4.2 Comparing Models

The aim of the experiment is to test whether there is a significant difference in the size of the search space between a formulation where there is no aggregation of variables, and two alternative formulations where paraphrase variants are used such that aggregation of variables can be carried out.

To do this, binary variable IP problems were randomly generated in the following manner. An initial problem was generated with an objective function and one constraint, of size 100 vari-

[3]There are, of course, many other possible referring expressions if lexical changes are allowed—for example, *the piece of clothing*—but in this paper, only syntactic paraphrases are considered.

ables. Each variable had coefficients randomly generated from a uniform distribution with the range [-10,10]; the constraint had equal chance of being $\leq$ or $\geq$, and the constraint constant was generated from a uniform distribution with lower limit zero and upper limit equal to the sum of the variable coefficients. Each coefficient also had associated with it a maximum flexibility representing the extent to which it could vary; this value was generated from a uniform distribution with the range [0,3]. This was then repeated with 2 and 3 constraints, with initial number of variables 400 and 2000 respectively.

Then, the variables were aggregated together using two different algorithms, POLE and BALANCE. The algorithms vary the coefficients where possible, choose a paraphrase variant whose c-set, for a particular referring expression, matches the c-set of at least one fixed-coefficient TARGET variable, and aggregate those two variables together. The difference between the algorithms is that, when faced with a choice between target variables, POLE chooses the one with the largest range (i.e., the one comprising the most aggregated variables) while BALANCE chooses the one with the smallest range. Any problems where the two algorithms gave the same aggregations were discarded for the purposes of differentiating between the two.

In order to construct problems that could be solved within a reasonable time, subproblems were extracted from these randomly generated problems: only those variables which took part in the aggregation (under either algorithm) were included in the subproblem. These subproblems then had an average of around 21 variables. The subproblems in the three formulations—the default no-aggregation, aggregation using the POLE algorithm, and aggregation using the BALANCE algorithm—were then solved using a branch-and-bound technique.

4.3 Results

The mean search space sizes (in number of nodes) and their standard deviations are shown in Table 2. Sample sizes are 1493, 421 and 44.[4]

		mean	std. dev.
one constraint	default	910.09	3616.7
	POLE	371.22	1792.8
	BALANCE	381.48	2037.4
two constraints	default	201.45	1435.5
	POLE	21.76	72.7
three constraints	default	2232.0	6682.7
	POLE	192.23	341.0

Table 2: Comparing search sizes of algorithms

It is clear that hypothesis tests assuming the normal distribution are not likely to be appropriate for testing difference between means—the distributions are very skewed, with standard deviations much larger than the mean, and all values positive—so the non-parametric Wilcoxon paired sign-rank test was used for comparison. For one constraint, the algorithms were pairwise compared: taking as the null hypothesis that the means were equal, the probabilities that this was the case were 1.18×10^{-5} (default vs. POLE), 1.07×10^{-4} (default vs. BALANCE), and 1.26×10^{-4} (POLE vs. BALANCE). For two and three constraints, only POLE (the better of the two algorithms) and the default were compared, giving probabilities of 1.74×10^{-21} and 7.63×10^{-5} respectively.

4.4 Discussion

The aggregate formulations are clearly significantly better than the default formulation in terms of expected search space size. As well, the aggregate formulations have a more predictable search size (a substantially smaller standard deviation), which is also desirable.

A preliminary look at applying the technique to an actual text produces similar results. A 12-sentence text was taken from *The Atlantic Monthly*, with 19 possible paraphrases for the text, and length and readability constraints set. Setting $k1$ (length change) at 0, and varying $k2$ (average sentence length) between 15 and 20, the search space for the POLE algorithm was always the same size as or smaller than for the default by the same order of magnitude as for the randomly generated data. One difference between the actual text and the experimental data is that the coefficients for actual text paraphrases are

[4]The sample sizes are unusual as the problems where the POLE and BALANCE formulations were the same were discarded.

not independent, unlike the randomly generated experimental examples. This meant that there were more aggregations than expected. Looking at how likely aggregations are to occur in actual text—as opposed to investigating how effective they are in cutting search space sizes, the aim of this paper—is future work.

Relating this to other work, there are parallels here with the concept of approximate symmetry (Ellman, 1993): the variants of a particular paraphrase are all close to each other (as referring expressions in general only vary to a small extent in syntactic paraphrasing), and can naturally be grouped into disjoint sets, one set per paraphrase type. However, the sets of paraphrase variants do not have the required entailments for Ellman's approximate symmetry.

Formulating the paraphrase problem as an IP allows an easy translation to using heuristic search methods. Genetic Algorithms (GAs) require encoding the problem as a 'chromosomal' string, which, given the formulation here, is merely the concatenation of the decision variables; crossover and mutation occur between and on these strings. Similarly, in Simulated Annealing (SA) the decision variables form the state of the annealing. The process of aggregation described in this paper complements, rather than attempts to replace, such heuristic methods. GAs and SA reduce search time by applying heuristics to the process of moving around the search space. However, the models themselves are the same size in terms of number of variables, with just the solution space navigated differently. Aggregation reduces the size of the model, in a way related to factor analysis (see Biber, 1988), by compressing variables. Both approaches can thus be used together, aggregation decreasing the number of variables, and GAs and SA heuristically navigating the already-reduced search space.

5 Conclusion

Existing systems dealing with paraphrase only look at constraints at the level of sentences or clauses. The search strategies they use are adequate for this scale of paraphrase; however, this is not the case where the problem is the real world situation of external constraints imposed on an entire text, particularly for industrial-sized applications. This paper has looked at a computational model of paraphrase for entire texts that can use optimisation techniques to cut the size of the search space to make the problem feasible. It then proposed a refinement of the model using ideas related to symmetry breaking to reduce the size of the model and hence to cut the search space further: aggregating paraphrases that, at the level of abstraction used for the model, can be made interchangeable by varying referring expressions or other aspects of particular paraphrase types. Simulations then confirmed that the model using aggregated paraphrases has a significantly smaller mean search space, and also a significantly smaller variance of search space size, than the basic model.

References

Beale, S. 1997. Using Branch-and-Bound with Constraint Satisfaction in Optimization Problems. *Proc. of AAAI97*, 209–214.

Biber, D. 1988. *Variation Across Speech and Writing.* Cambridge Univ. Press. Cambridge, UK.

Dras, M. 1997. Reluctant Paraphrase: Textual Restructuring under an Optimisation Model *Proc. of PacLing97*, 98–104.

Dras, M. 1997. Representing Paraphrases Using STAGs. *Proc. of ACL-EACL97*, 516–518.

Ellman, T. 1993. Abstraction via Approximate Symmetry. *Proc. of the 13th IJCAI*, 916–921.

Inui, K., T. Tokunaga and H. Tanaka. 1992. Text Revision: A Model and its Implementation. *Proc. 6th Internat. Workshop on NLG*, 215–230.

Jordan, M. 1994. Toward Plain Language: A Guide to Paraphrasing Complex Noun Phrases. *J. of Technical Writing and Communication*, 24(1), 77–96.

Joshi, A., L. Levy and M. Takahashi. 1975. Tree Adjunct Grammars. *J. of Computer and System Sciences*, 10(1).

Joslin, D. and A. Roy. 1997. Exploiting Symmetry in Lifted CSPs. *Proc. of AAAI97*, 197–202.

Kieras, D. 1990. *The Computerized Comprehensibility System Maintainer's Guide.* Michigan Univ. Rpt. 33.

Klare, G. 1974–75. Assessing Readability. *Reading Research Quarterly, Number 1, 1974–1975*, 62–102.

Nemhauser, G. and L. Wolsey. 1988. *Integer and Combinatorial Optimization.* Wiley & Sons. NY, NY.

Robin, J. 1994. *Revision-Based Generation of Natural Language Summaries Providing Historical Background.* Columbia Univ. Report CUCS-034-94.

Williams, H.P. 1995. *Model Building in Mathematical Programming.* Wiley and Sons. Chichester, UK.