Automatic generation of large-scale paraphrases

Richard Power and **Donia Scott**
Centre for Research in Computing
The Open University
Walton Hall
Milton Keynes MK7 6AA
{r.power,d.scott}@open.ac.uk

Abstract

Research on paraphrase has mostly focussed on lexical or syntactic variation within individual sentences. Our concern is with larger-scale paraphrases, from multiple sentences or paragraphs to entire documents. In this paper we address the problem of *generating* paraphrases of large chunks of texts. We ground our discussion through a worked example of extending an existing NLG system to accept as input a source text, and to generate a range of fluent semantically-equivalent alternatives, varying not only at the lexical and syntactic levels, but also in document structure and layout.

1 Introduction

Much work on paraphrase generation has focussed on lexical variation and syntactic transformation within individual sentences (Barzilay and McKeown, 2001; Carroll et al., 1999; Dras, 1999; Inui and Nogami, 2001; Kozlowski et al., 2003; Langkilde and Knight, 1998; Takahashi et al., 2001; Stede, 1999). Our interest in this paper lies instead with variations at the level of text structuring — the way in which propositions are grouped into units like paragraphs, sections, and bulletted lists, and linked rhetorically by discourse connectives such as 'since', 'nevertheless', and 'however'. Elsewhere, we have described a text-structuring method in which the options for organising propositions in a text are laid out as a set of constraints, so that acceptable solutions can be enumerated using constraint satisfaction and evaluated using a cost metric (Power et al., 2003). In this paper we show how this method, when harnessed to a system for recognising rhetorical structure in an input text, can be employed in order to produce large-scale paraphrases fulfilling purposes like improving coherence and achieving a desired style of layout.

2 Text structure

The input to our text-structuring system (ICONOCLAST) is a rhetorical structure tree (Mann and Thompson, 1983) in which the leaves are elementary propositions, specified either as semantic formulas or as canned text. The following is a simple example, containing one nucleus-satellite relation (REASON) and one multinuclear relation (CONJUNCTION[1]):

```
reason
NUCLEUS: recommend(doctors, elixir)
SATELLITE: conjunction
  1: quick-results(elixir)
  2: few-side-effects(elixir)
```

Ignoring variations in the wording of propositions, ICONOCLAST generates over 20 texts realising this input (or many more if a larger repertoire of discourse connectives is allowed). They include the following two solutions, which lie at stylistic extremes, the first compressing the message into a sentence (suitable if space is at a premium), the second laying it out more expansively in a list:

[1]This is the RST relation, LIST, which we have renamed here to avoid possible confusion with the layout style of vertical lists.

Solution 1
Doctors recommend Elixir since it gives quick results and it has few side effects.

Solution 2
- Elixir gives quick results.
- Elixir has few side effects.

Therefore, it is recommended by doctors.

Comparing these solutions illustrates some of the text-structuring options, and some ways in which they interact.

- Propositions, or groups of propositions, can be realised by different *text categories*. Thus `quick-results(elixir)` is realised by a text-phrase (in Nunberg's sense (Nunberg, 1990)) in Solution 1, and by a text-sentence (also a list item) in Solution 2.

- Rhetorical relations can be expressed by different discourse connectives or layout options. In Solution 1, REASON is realised by 'since' and CONJUNCTION by 'and'; in Solution 2 REASON is realised by 'therefore' and CONJUNCTION (more implicitly) by a bulletted list.

- Propositions may be realised in different orders: for instance, the nucleus of the REASON relation comes first in Solution 1, while the satellite comes first in Solution 2. Note that order is constrained by the choice of discourse connective: 'therefore' requires satellite-first; 'since' allows both orders.

These text-structuring decisions strongly influence the options for wording the individual propositions, mostly because they determine the order in which propositions are presented. In Solution 1, the nucleus of the REASON relation is presented first, so 'Elixir' has to be referenced by name, and there is no particular reason for preferring passive to active. In Solution 2, the same proposition occurs at the end, when Elixir has been introduced and established as the topic focus; it is therefore appropriate to refer to Elixir by a pronoun, and to promote this reference to the most salient position in the clause by passivization.

Text structuring is controlled by *hard constraints*, which determine the set of solutions that can be generated, and by *preferences* (or soft constraints), which allow a ranking of solutions from best to worst. The purpose of hard constraints is to avoid solutions that are clearly anomalous, such as the following text in which the arguments of the CONJUNCTION relation are separated, thus altering the rhetorical interpretation:

> Since Elixir gives quick results doctors recommend it, and it has few side effects.

A more marginal case is the following solution, in which the arguments of a nucleus-satellite relation are expressed as items in a bulletted list. In the default settings this is also considered an anomaly, since a bulletted list usually implies a parallelism among the items that is violated when one argument dominates the other.

- Elixir gives quick results and has few side-effects.
- Therefore, it is recommended by doctors.

The purpose of soft constraints is to represent stylistic preferences. These include general principles of prose quality that are likely to apply to any context, as well as preferences specifically linked to the purpose of the text and the nature of the intended reader. Here are four examples of preferences supported in ICONOCLAST: we would assume that the first two are general, the second two specific.

- **Avoid single-sentence paragraphs** This would penalise a solution in which our example was laid out in two paragraphs, one for satellite and one for nucleus.

- **Avoid discontinuity of reference** As Kibble and Power (2004) have shown, centering criteria can be used to penalize solutions with relatively many topic shifts.

- **Avoid passivization** In contexts requiring an informal, popular style, there might be a stronger tendency to favour active over passive.

- **Avoid complex sentences** For some contexts we might prefer to penalize solutions in which many propositions are presented within the same sentence (e.g., Solution 1).

All these preferences are implemented through a cost metric. To calculate the cost of a solution,

the program first recognizes all violations, then multiplies each by a weight representing its importance before summing to obtain a total score. During execution, the program can either enumerate all solutions, ranking them from low cost to high, or it can simply search for the best solution using branch-and-bound optimization.

3 Controlling constraints and preferences

ICONOCLAST was originally developed as a component of a Natural Language Generation system. It assumes that the propositional content of the desired text is already formally encoded, along with a rhetorical-structure tree representing the role of each proposition in the argument. The program can also be run on a simplified input in which propositions are replaced by canned phrases; however, the quality in this case will obviously suffer, since referring expressions and clause structure cannot be adapted to context. By itself, then, ICONOCLAST cannot be used in order to paraphrase an independently provided text. However, once a semantic model is available, the system allows an unusual degree of flexibility and precision in controlling paraphrases. The source of this power lies in the use of explicitly encoded constraints and preferences, which can be edited through a direct-manipulation user interface in order to guide the generator in the desired directions.

For *hard constraints*, the control interface works mostly by buttons for switching constraints on and off, or occasionally by menus for fixing the value of a parameter. Examples of switches are the following (also mentioned above):

> Allow indented list for arguments of a multinuclear relation (Yes/No)
>
> Allow indented list for arguments of a nucleus-satellite relation (No/Yes)
>
> Allow discourse connective introducing a list item (Yes/No)

The default in each case is the option given first, which would allow (but not require) a solution to our example in which the conjunction was realised by a list including the discourse connective 'and':

> Doctors recommend Elixir because
> • Elixir gives quick results
> • And it has few side effects

An example of a parameter setting would be a constraint fixing the textual unit governing the whole text, or the maximum text level allowed for an indented list item:

> Root textual unit
> (document/section/paragraph/text-sentence)
>
> Maximum level for list item
> (paragraph/text-sentence/text-clause)

By constraining the whole text to fit in a paragraph, we could eliminate any solution requiring multiple paragraphs (e.g., nucleus in one paragraph and satellite in another). Under this setting, both solutions 1 and 2 could be generated (although solution 1 would have to be a single-sentence paragraph). Further constraining the root level to `sentence` would preserve solution 1 but eliminate solution 2.

For *soft constraints*, the user interface works through sliders representing both the direction of a preference and its intensity. In most cases, the sliders are calibrated to an 11-point scale from -5 to +5. A straightforward example is the dichotomy between active and passive voice, where negative values penalize use of the passive, while positive values penalize use of the active; the central value (zero) represents neutrality. A cost value is computed every time a proposition is realised by a clause for which the grammar allows passivization. Depending on the setting of the 11-point scale, a cost is incurred either for use of the passive (negative values on the scale), or for failure to use it (positive values on the scale); the amount of cost varies from 1 to 5, again depending on the setting. Thus if the user sets the passivization slider to a value of -4, a cost of 4 accrues every time a proposition is realised by a passive clause; or for a value of +2, a cost of 2 accrues every time a proposition that could have been realised by a passive clause is realised by an active one.

In practice, this method of evaluating solutions typically means that *every solution is flawed*, given a non-trivial semantic input and a sufficient range of preferences. The reason is that many decisions are trade-offs: avoiding cost on

one preference often means incurring cost elsewhere. For instance, a preference to avoid the passive conflicts with the preference to preserve topical coherence, which is expressed by penalizing a 'salience violation' — that is, a failure to equate the backward-looking center in a clause with the most salient forward-looking center (i.e., Cb with Cp) (Kibble and Power, 2004). If salience requires passivization, and passivization is penalized, then a cost must be incurred somewhere: the issue is which is the lesser evil.

We have considered two ways of controlling a paraphrase in a constraint-based generator: imposing/relaxing a hard constraint, and changing a preference. A possibility that we have not yet implemented is a hard constraint defined only *on the current problem*, as opposed to the general settings illustrated above. The constraint might state, for example, that the proposition `recommend(doctors, elixir)` should appear at the beginning of the text, thus eliminating Solution 2. Or it might state that the conjunction relation between the other propositions should be realised by a bulletted list, thus eliminating Solution 1. To support constraints of this kind one would need a user interface in which the user can select part of the semantic input, perhaps by clicking on the corresponding part of the text, as in a WYSIWYM interface (Power and Scott, 1998); a dialogue box would then appear allowing a range of constraints specifically directed to the selected fragment. Such an interface would mimic the typical interaction between a human writer and human critic — e.g., the critic might highlight a paragraph and advise the writer to reformat it as a list.

4 Deriving the rhetorical-semantic input

We have shown that by defining text-structuring as a Constraint Satisfaction Problem, our method allows considerable flexibility and precision in controlling the generation of paraphrases (Power et al., 2003). The question now is whether the system can be extended so that it accepts a text as input, rather than a formally encoded rhetorical-semantic representation. Obviously the extended system will require an extra component performing interpretation of the input text — but how much interpretation is needed in order to provide an encoding that the current ICONOCLAST text-structurer can use? Can we extract sufficient rhetorical and referential information to allow reasonable paraphrases, without depending on a full semantic analysis of the original text?

In this section we consider three stages of interpretation, which could be applied incrementally:

1. *Rhetorical mark-up:* The program marks up the EDUs (Elementary Discourse Units) (Marcu, 2000) in the input text — what we have been calling the elementary propositions — and also identifies the rhetorical relations among them, expressed through a Rhetorical Structure Tree. Within EDUs there is no mark-up: at this stage they are treated as canned strings.

2. *Coreference mark-up:* The program identifies noun-phrases referring to discourse entities, and recognises chains referring to the same entity. For each discourse entity, enough semantic information is recovered to allow a correct choice of pronoun (i.e., values are assigned to features like NUMBER, GENDER, HUMAN), but no further semantic analysis is assumed.

3. *Clause transformations:* The syntactic structure of each EDU is analysed sufficiently to allow a reformulation that promotes a different discourse entity as the most salient of the clause (i.e., the Cp). Typically this would mean a change of voice from active to passive, or vice-versa, although there might be other variations like fronting that could be explored.

We now discuss these stages in turn.

4.1 Recognising rhetorical structure

Maintaining the same example, suppose that the input text is the following (a slight variation of Solution 1):

> Doctors recommend Elixir since it gives quick results and has few side effects.

The goal at this stage is to interpret this text as a set of elementary propositions, represented by canned phrases, organised into a tree by rhetorical relations. An example of the desired encoding, in

the format actually used as input to the current system, is the following XML fragment:

```
<RhetRep relation=reason>
  <SemRep prop="doctors recommend Elixir"/>
  <RhetRep relation=conjunction>
    <SemRep prop="it gives quick results"/>
    <SemRep prop="it has few side-effects"/>
  </RhetRep>
</RhetRep>
```

As can be seen, even though this representation provides no analysis within propositions (EDUs), the task of deriving the rhetorical structure and the canned phrases is not trivial. First, the rhetorical relations REASON and CONJUNCTION must be inferred. Second, the correct tree structure must be assigned, with REASON dominating CONJUNCTION. Third, the discourse connectives 'since' and 'and' must be separated from the phrases in which they occur — the aim is that these phrases should represent only the propositions. Finally, where parts of a phrase have been elided through aggregation (e.g., 'has few side-effects'), the missing part ('it') should be found and replaced.

If this level of interpretation is achieved, the program will be able to generate several dozen paraphrases, but referential continuity will be poor unless we pose the additional constraint that the order of propositions should remain the same as in the original. Thus a successful paraphrase, including some reformatting, would be the following:

> Doctors recommend Elixir since:
> • it gives quick results.
> • it has few side effects.

However, with satellite preceding nucleus, as in Solution 2, the text becomes incoherent because the first mentions of Elixir are through a pronoun.

> • It gives quick results.
> • It has few side effects.
> Therefore, doctors recommend Elixir.

4.2 Recognising coreference

Incoherence resulting from canned propositions can be partly remedied if the analysis of the input text is taken a stage further, by recognising some simple semantic features on noun phrases, and marking them up for coreference. The elementary propositions in our example could for instance be marked up as follows:

```
<edu>
  <np id=1 phrase="doctors"
      class="human" number="plural"/>
  recommend
  <np id=2 phrase="Elixir"
      class="thing" number="singular"/>
</edu>
<edu>
  <pronoun id=2 phrase="it"/>
  gives
  <np id=3 phrase="quick results"
      class="thing" number="plural"/>
</edu>
<edu>
  <pronoun id=2 phrase="it"/>
  has
  <np id=4 phrase="few side-effects"
      class="thing" number="plural"/>
</edu>
```

This further mark-up facilitates text-structuring in two ways. First, since centering information is now available (the Cb and Cp of each proposition can be computed), the evaluation of solutions can take account of the centering preferences proposed by Kibble and Power (2004). Secondly, when realising individual propositions, the referring expressions can be adapted to context, perhaps by replacing a name/description with a pronoun, or even eliding it altogether when two propositions are aggregated. This means that the program will be able to generate solutions such as the following, in which the wordings of the propositions has been revised:

> Since Elixir gives quick results and has few side-effects, doctors recommend it.

This solution illustrates three ways in of revising a proposition:

- $Pronoun \rightarrow Name$ 'it gives quick results' becomes 'Elixir gives quick results'.
- $Elision$ 'it has few side-effects' becomes 'has few side-effects'.
- $Name \rightarrow Pronoun$ 'doctors recommend Elixir' becomes 'doctors recommend it'.

The generated paraphrases should now be more fluent, but the program is still limited by its inability to control the most salient referent in a proposition (i.e., to modify the Cp). To add this option, we need the third level of interpretation mentioned above, in which the structure of a clause can be transformed (e.g., from active to passive).

4.3 Clause transformations

Assuming that the analysis program can completely parse a clause identified as an EDU, it may be able to apply a syntactic transformation which expresses the same proposition with changed information focus. An obviously useful transformation is passivization — or its opposite if the original sentence is in the passive. Assuming that the parser has correctly identified the main verb, and that the program has access to a lexical database including irregular morphology, it could derive alternative formulations for the original propositions by a rule such as the following:

```
[NP1] recommends [NP2]⇒
[NP2] is recommended by [NP1]
```

Of course the program should not allow such transformations for special verbs like 'be' and 'have', so as to avoid clumsy renderings like 'few side effects are had by Elixir'. However, when used on an appropriate verb, passivization can improve the fluency of the solution by promoting the Cb of the proposition to the subject position, so that it becomes the Cp; revisions of this kind also provide more opportunities for favourable pronominalization and elision. With this extra resource, the solution just proposed can be improved as follows:

> Since Elixir gives quick results and has few side-effects, it is recommended by doctors.

A more ambitious aim would be to transform between finite and reduced forms of a subordinate clause. For instance, if the original text is 'Despite having few side-effects, Elixir is banned by the FDA', we could allow the transformation of 'having few side-effects' into the finite clause 'Elixir has few side-effects, borrowing the subject and tense from the main clause. This transformation would enable the system to generate a solution using a connective such as 'however' which requires that full clauses are employed both for the nucleus and the satellite. Alternatively, a finite clause could be transformed into the reduced form, so allowing the connective 'despite'.

Conclusion

It is hard to conceive of an NLG system that cannot produce alternative realisations, and thus paraphrases. Most systems, however, are only capable of producing variations at the lexical or syntactic levels (or both). As such, they operate very much like traditional Machine Translation systems — except that the source and target texts are now in the same language — and have similar limitations. Additionally, most of them work with input that is a representation of the meaning of a (source) text, rather than the text itself.

The system described in this paper develops an existing NLG system into a full-blown paraphase generator capable of producing a wide range of alternative renditions of the source text, with variations at three linguistic levels: lexical choice, syntactic structure, and document structure. This is in contrast to most existing paraphrase generators, which are constrained to vary only the first or second of these levels (Barzilay and McKeown, 2001; Carroll et al., 1999; Dras, 1999; Inui and Nogami, 2001; Kozlowski et al., 2003; Langkilde and Knight, 1998; Takahashi et al., 2001; Stede, 1999). The range of lexical and syntactic variation in a paraphrase generator obviously depends on how deeply the input text is interpreted, but even with the relatively superficial analysis proposed here, we can introduce variations for discourse connectives, referring expressions (in particular, when to use pronouns), and some clause patterns (e.g., whether to use active or passive). However, the innovation in our work lies in its controlled variation in the third level, document structure: just as the other paraphrase generators provide multiple lexical-syntactic structures for the same semantic structure, so our system provides multiple document structures for the same discourse structure (i.e., for the same rhetorical structure). The document structure solutions serve not only to realise the rhetorical input, but also to create a context that determines which of the alternative syntactic realisations is most suitable for the elementary propositions.

Our paraphrase generator links an existing general-purpose discourse parser — DAS (Le Thanh et al., 2004)[2] — which builds a discourse tree automatically from an input text, to an existing NLG system —ICONOCLAST (Power et al., 2003) — which generates a wide range of

[2]Similar parsers have been developed by Marcu (2000) and Corston-Oliver (1998)

formulations for a given discourse structure. We have described here the issues that need to be taken into account when turning any NLG system into a fully-fledged paraphraser. We believe that our approach to text-structuring, whereby options for organising propositions in a text are laid out as a set of constraints, and acceptable solutions are enumerated using constraint satisfaction and evaluated using a cost metric, provides a particularly useful method for achieving large-scale paraphrases. Although we are agnostic with respect to the issue of psychological validity, it is worth noting that our method reflects many of the processes facing any writer or editor trying to achieve their ideal text, but constrained by the linguistic resources at hand (e.g., wording, syntax, discourse and layout) which interact with each other such that the final text is invariably a flawed version of the ideal.

For evaluation of our system, two points need to be addressed. The first concerns *fidelity*: are the generated solutions equivalent in meaning to the original input text? The second concerns *quality*: are the generated solutions ranked, by the cost metric, in a way that corresponds to the preferences of good judges? More practically, we would like to explore the issue of *usability*: the main question here is whether human users can successfully manipulate the system's constraints and preferences in order to guide solutions in the desired direction.

References

Regina Barzilay and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 50–57, Toulouse.

J. Carroll, G. G. Minnen, D. Pearce, Y. Canning, S. Devlin, and J. Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL-99)*, pages 269–270, Bergen, Norway.

S. Corston-Oliver. 1998. *Computing Representations of the Structure of Written Discourse*. Ph.D. thesis, University of California, Santa Barbara, CA, USA.

Mark Dras. 1999. *Tree Adjoining Grammar and the Reluctant Paraphrasing of Text*. Ph.D. thesis, Macquarie University, Australia.

Kentaro Inui and Masaru Nogami. 2001. A paraphrase-based exploration of cohesiveness criteria. In *Proceedings of the 8th European Workshop on Natural Language Generation (EWNLG-01)*.

Rodger Kibble and Richard Power. 2004. Optimising referential coherence in text generation. *Computational Linguistics*, 30(4).

Raymond Kozlowski, Kathleen F. McCoy, and K. Vijay-Shanker. 2003. Generation of single-sentence paraphrases from predicate/argument structure using lexico-grammatical resources. In *Proceedings of the Second International Workshop on Paraphrasing*, pages 1–8.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL98)*, pages 704–710, Montreal.

H. Le Thanh, G. Abeysinghe, and C. Huyck. 2004. Generating discourse structures for written texts. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-2004)*, pages 329–335.

W.C Mann and S.A. Thompson. 1983. Relational propositions in discourse. Technical Report RR-83-115, Information Sciences Institute.

D. Marcu. 2000. *The theory and practice of discourse parsing and summarisation*. MIT Press, Cambridge, Massachusetts, USA.

Geoffrey Nunberg. 1990. *The Linguistics of Punctuation*. CSLI Lecture Notes, No. 18. Center for the Study of Language and Information, Stanford.

Richard Power and Donia Scott. 1998. Multilingual authoring using feedback texts. In *Proceedings of 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 98)*, pages 1053–1059, Montreal, Canada.

Richard Power, Donia Scott, and Nadjet Bouayad-Agha. 2003. Document structure. *Computational Linguistics*, 29(2):211–260.

Manfred Stede. 1999. *Lexical semantics and knowledge representation in multilingual text generation*. Kluwer Academic Publishers, Boston.

Tetsuro Takahashi, Tomoyam Iwakura, Ryu Iida, Atsushi Fujita, and Kentaro Inui. 2001. Kura: A transfer- based lexico-structural paraphrasing engine. In *Proceedings of the Workshop on Automatic Paraphrasing. (NLPRS 2001)*, Tokyo.