Target suites for evaluating the coverage of text generators

John A. Bateman[*], **Anthony F. Hartley**[†]

[*]Sprach- und Literaturwissenschaften
Universitaet Bremen
bateman@uni-bremen.de

[†]Information Technology Research Institute
University of Brighton, UK
Tony.Hartley@itri.brighton.ac.uk

Abstract

Our goal is to evaluate the grammatical coverage of the surface realization component of a natural language generation system by means of *target suites*. We consider the utility of re-using for this purpose test suites designed to assess the coverage of natural language analysis / understanding systems. We find that they are of some interest, in helping inter-system comparisons and in providing an essential link to annotated corpora. But they have limitations. First, they contain a high proportion of ill-formed items which are inappropriate as targets for generation. Second, they omit phenomena such as discourse markers which are key issues in text production. We illustrate a partial remedy for this situation in the form of a text generator that annotates its own output to an externally specified standard, the TSNLP scheme.

1. Test suites and target suites

The evaluation of natural language generation (NLG) systems is an issue that few authors have addressed seriously to date. Among the principal reasons is the difficulty of defining what the input should be and, indeed, of assessing the quality of the output (cf., (Sparck-Jones and Galliers, 1996)). Evaluations conducted to assess the adequacy of a system for a particular application have tended to focus on the quality of the output text, particularly its fluency or intelligibility (cf., (Coch, 1996; Lester and Porter, 1997)). While accepting that evaluation will inevitably bear on the output, (Mellish and Dale, 1998) suggest a finer-grained approach which aims to tease out the contribution made by a particular sub-task in the generation process to the quality of the final output. We propose here to take up that suggestion and to consider the final task in the line–surface realization.

There is a general recognition that a crucial question about a surface realization is its grammatical coverage (cf. (van Noord and Neumann, 1997; Mellish and Dale, 1998)). This question poses itself not only for adequacy evaluation intended to assess applications potential, but also diagnostic and progress evaluation. In other words, it is relevant at all stages in a system's life-cycle, from conception to fielding.

A favoured means of gauging the coverage of NLP systems designed for analysis tasks is test suites, which consist of controlled and systematically organized data (cf., (Lehmann et al., 1996; Oepen, Netter and Klein, 1997; Netter et al., 1998), in contrast to naturally occurring text corpora. It is, then, sensible to see whether any of these existing suites can be re-used or re-purposed for the benefit of systems designed for generation.

Test suites as currently conceived are intended as input data. For some tasks where the output can be well-defined – such as message or speech understanding – test collections provide both input and output data. For NLG evaluation, we propose the use of *target suites* that specify a useful set of outputs for a generator (typically sentences showing particular syntactic structures), while remaining agnostic on the inputs. Practically, however, particular generation systems are then encouraged to provide corresponding inputs so that both coverage and structural treatments can be readily compared. A first step towards this is reported in (Henschel, Bateman and Matthiessen, 1999) for an extensive set of nominal referring expressions; input specifications are provided for two of the broadest coverage generators for English currently available–the KPML/Nigel and FUF/Surge generators. We are now developing further target suites both for other languages and for other areas of grammar, using the mechanism described below.

Two questions arise. Which of the test items already available are relevant target items for NLG? What phenomena important for NLG are missing and need therefore to be added to the target suites in order to cater for this application? This second question is addressed later in the paper. We have taken as our reference here the TSNLP test suites (Lehmann et al., 1996; Oepen, Netter and Klein, 1997). Paradoxically, the relevance for NLG of the items in the TSNLP suite turns out to be diminished by both their generality and their application-specificity.

On the one hand, the compilers of the suite were motivated by the desire to ensure that the data should be reusable and not tailored to a single type of application (Estival et al., 1995). Thus, they mostly applied general guidelines for creating the data set, although they did not adhere to them in all cases. These guidelines strongly favour, therefore, unmarked word order and declarative, active, indicative sentences in the present tense with a 3rd person singular subject–to mention just a few constraints which mean that any NLG system would be seriously under-exercised by the suite in its current state.

On the other hand, the designers targeted three specific applications–parsers, grammar checkers and controlled language checkers–whose evaluation requires ungrammatical data not systematically available in text corpora. Indeed, over 35% of the 14817 English, French and German items in the TSNLP suite are ungrammatical and as such irrelevant as target items for NLG. A more recent project aimed at producing a more efficient environment for test suite construction, DiET (Netter et al., 1998), is validating its data on checkers, MT systems and translation memories. While referring to 'NLP systems' generally, the researchers make no mention of NLG applications.

In the next section we describe the implementation of a novel approach to producing resources for assessing the coverage of NLG systems.

2. Implementation of automatic annotation

A key feature of all test suites is a suitable annotation scheme that allows extraction of sets of items appropriate to particular evaluation goals. The fact that NLG is goal- and content-driven requires a scheme that relates items not only formally but also semantically. In this section, we show how an existing text generator–KPML (Bateman, 1997)– has been adapted so as to automatically annotate its own output with an externally specified annotation scheme–provided by the TSNLP project (Oepen, Netter and Klein, 1997). Such annotations, which are both formal and functional in nature, make it easier to assess the coverage of the generator and to compare its coverage with that of other systems. Moreover, these annotations can provide an essential link to corpora, insofar as they establish–using a widely understood metalanguage–links between suite items and domain-specific corpora (cf. (Netter et al., 1998)).

We propose that such facilities should become a standard part of generator design in order to ensure reliability and appropriate documentation of coverage.

As we have noted, work on test suite design has been oriented, until now, to the evaluation of NLP tools for analysis. Nevertheless, many classifications developed in that work remain useful for generation. In addition, the adoption by the NLG community of test suite categories that are already to some extent in use within the Language Engineering and NLP communities will also aid comparability between linguistic resources designed for analysis–such as TS-GRAM–and those designed for generation–such as FUF/Surge (Elhadad and Robin, 1996), KPML/Nigel (Bateman, 1997) and RealPro (Lavoie and Rambow, 1997).

2.1. Exploiting the generation history

There is an interesting difference between the construction of test suites and the construction of target suites, as we have noted above. Whereas the items included in test suites need to be first selected and then marked up, the test items in target suites are *themselves being generated*. This is significant since we can then make use of the complete history of the construction of any item. This history includes the grammatical constituency structure, as well as all of the semantic-functional decisions that were taken in order to reach the result. It is then possible that a substantial proportion of the information needed to create a properly annotated test suite item is *already* available somewhere in the generation history.

Indeed, this opportunity has long been exploited in the development of NLG systems. Most large-scale systems provide sets of examples that show input-output pairs. Even if the inputs to different systems ranges from high-level semantic specifications, via abstract syntactic structures to rich syntactic specifications (cf. (Mellish and Dale, 1998)), the act of making them systematically available permits comparisons that would otherwise be obscured. The output is, minimally, a string corresponding to the input specification; we show below how it can be enriched as a by-product of the generation process.

One of the earliest such sets that was intended to show definitively the coverage of a grammar was the "Exercise Set" developed for the Nigel grammar of English within the Penman generation system. In current environments for the development of generation grammars, such as KPML, the role of the example set has been enhanced to make it a major development tool. Users seeking to extend linguistic resources typically work from examples that come close to their required output. Then, since the generation system actually generates the strings given in the example, the complete decision path and resulting grammatical structures created for that example are open to inspection. The example therefore indexes precisely those resources that have been activated in the course of generating that particular example. This information can then serve directly to provide automatic target suite annotation.

As an example that will run throughtout this section, consider the following simple sentence from the TSNLP documentation (Estival et al., 1995)–page 42:

Has he carried it?

This is an example of the TSNLP phenomenon type S_Types-Questions-Y/N-questions-Inverted. It serves to test inversion of subjects and finite verbs in polarity (yes/no) questions.

When generating such a sentence, we must provide for our chosen NLG generator an appropriate input. For the KPML/Nigel generator that we have extended to support automatic annotation, such inputs are given in the Sentence Plan Language (SPL) notation developed by Kasper, R., 1989). An appropriate input for our running example is shown in Figure 1. Note that this is the full unabbreviated input–typically users make use of numerous macros and defaults in order to simplify their input. In this example, most of the complexity comes from the complex temporal relationships that are necessary to fully motivate a past perfect tense in English. When this SPL specification is passed to the KPML tactical generator with the Nigel English grammar loaded, the corresponding string (and only this string) is produced.

Standard debugging and resource maintainance tools within KPML do considerably more than just produce the string however.

- First, as developed within the Penman generation system, the generated result may be stored to an *example record*. Example records are the basis of the exercise sets mentioned above. Each example record includes the semantic input, the grammatical features selected

```
(e0 / (carry directed-action)
  :actor (x1 / male
    :identifiability-q identifiable
    :empty-number-q empty)
  :actee (x2 / object
    :identifiability-q identifiable
    :empty-number-q empty)
  :speech-act-id (sa / question
    :polarity variable
    :speaking-time-id (st / time
      :time-in-relation-to-speaking-time-id
        (rt / time
          :precede-q (rt st) notprecedes
          :precede-q (rt et) notprecedes)
      :time-in-relation-id (st et rt) et
          :precede-q (st rt) notprecedes))
  :event-time (et / time
          :precede-q (et rt) precedes))
```

Figure 1: Full KPML/Nigel semantic input for "Has he carried it?"

from the grammar during generation, the semantic decisions made, and the grammatical structure generated. As with most systemic-functional generators, it is the list of grammatical features selected that completely determines the structure built. When the running example is generated by the Nigel grammar, the clause constituent is seen to be the result of 68 grammatical feature selections and the two nominal phrases ('he' and 'it') of 24 feature selections each. Maintaining these lists of features is crucial for using the example record as an index into the Nigel grammar, and has now proved itself to be a highly effective aid to the development of linguistic resources.

- Second, within KPML the generated string is not the direct result of generation but rather a further result of linearizing an internal annotated constituency structure (cf., (Bateman, 1999)). This constituency structure is used in KPML for presenting the generated string to developers in a mouse-sensitive fashion: each constituent of the generated string is individually mouse-sensitive so that the information from the example record can be accessed by direct manipulation.

These features of KPML have allowed us to implement automatic annotation very simply. The internal annotated constituency structure is no longer used to produce a recursive mouse-sensitive rendering of the generated string on-screen. Instead, it serves to print a direct XML version of that constituency structure to an output file.[1] Also, instead of making the selected grammatical features available via a mouse-click, the generator now folds this information directly into the XML expression as an attribute value. The result for the running example is shown in Figure 2, with the feature lists abbreviated for expository reasons. So far, this is merely a reformatting of information already maintained and managed during generation. Any example generated by the grammar with respect to any input semantics can be produced in such a form.[2]

[1]There were already methods in KPML for producing HTML-marked up strings as generated results, so this addition was trivial.

```
<UNIT
  class="CLAUSE"
  functions="SENTENCE"
  features=
    "CLAUSES CLAUSE FULL CLAUSE-SIMPLEX
     MOOD-UNIT NONCONJUNCTED
     INDEPENDENT-CLAUSE
     INDEPENDENT-CLAUSE-SIMPLEX
     NONINTERNAL-SUBJECT-MATTER
     INDICATIVE FINITE-CLAUSE
     FINITE-INSERTED TEMPORAL
     POSITIVE POSITIVE-FINITE
     PRESENT SECONDARY
     ...
     TEMPO0TEMPO1 DO-NEEDING-VERBS
     LEXICAL-VERB-TERM-RESOLUTION">
  Has
  <UNIT
    class="NOMINAL-GROUP"
    functions="SUBJECT TOPICAL ACTOR"
    features=
      "GROUPS-PHRASES GROUPS
       NOMINAL-LIKE-GROUPS
       NOMINAL-GROUP NONWH-NOMINAL
       NONEXTENDING-NOMINAL-GROUP-COMPLEX
       ...
       NONSUPERLATIVE NOMINATIVE
       HE-PRONOUN">
    he
  </UNIT>
  carried
  <UNIT
    class="NOMINAL-GROUP"
    functions="DIRECT-OBJECT GOAL"
    features=
      "GROUPS-PHRASES GROUPS
       NOMINAL-LIKE-GROUPS
       NOMINAL-GROUP NONWH-NOMINAL
       NONEXTENDING-NOMINAL-GROUP-COMPLEX
       ...
       NONSUPERLATIVE OBLIQUE
       NONDEMONSTRATIVE-SPECIFIC-PRONOUN">
    it?
  </UNIT>
</UNIT>
```

Figure 2: Direct XML-style translation of KPML example record for the example sentence

2.2. Mapping to externally specified annotations

Maintaining information about the grammatical units in terms of the grammatical features of the particular grammar used for generation does not, of course, support cross-system comparisons of coverage. Therefore the next

[2]The precise degree of constituency constructed can be manipulated in various ways: it is not necessarily a one-to-one re-rendering of the syntactic structure. This is described in the KPML documentation (Bateman, 1999) and is not directly relevant here however.

```
<UNIT class="CLAUSE"
  features=
    "S_TYPES-QUESTIONS-Y/N-QUESTIONS-INVERTED
     C_COMPLEMENTATION-DIVALENT-DIRECT_OBJECT"
  origin="generated"
  id=spl702>
Has he carried it?
</UNIT>
```

Figure 4: Automatically generated annotated example with TSNLP categories

stage in automatic annotation is to define mappings between the features of the generation grammar employed and standard classification features. Taking the TSNLP category for our running example, S_Types-Questions-Y/N-questions-Inverted, we must define a mapping between this and the generator-specific grammatical features shown in Figure 2. This complexity of this mapping will vary depending on the particular grammar used; examples of relatively complex mappings between TSNLP and an HPSG-based grammar are given, for example, in (Oepen, Netter and Klein, 1997).

Interestingly, however, the categories adopted in TSNLP appear to be more simply related to the categories of a generation grammar. The reasons for this are still unclear, although there is often a 'functional' flavour to the TSNLP categories reminiscent of much generation work. It may therefore be the case that generation grammars are more directly relatable to such test suite categories than the analysis grammars for which they were originally developed.

For our running example, it is relatively straightforward to find those features of the generated grammatical description that correspond to the TSNLP category, namely: {clause interrogative yes-no}. At present, we define a set of mappings from the generator-specific features to the target annotation scheme categories. Examples of mappings are given in Table 1. In future, it may be desirable to extend this; for example, since both the generator-specific grammatical features and the TSNLP terms are organised into type hierarchies, we may be able to use this for less exact matching.

The final stage is then to amend the XML-style output so that instead of the generator-specific feature lists, it gives all applicable standardized annotations. For grammatical units without a TSNLP correspondence, no markup is produced. For the running example and the mapping table given, the TSNLP-annotated XML-style expression is then as shown in Figure 4. The Figure also includes some of the other attributes that contribute to a test item, although we do not show the full structure here. We would also in general provide similar mappings from the grammatical 'function' categories shown in Figure 2 to TSNLP functors. This test item may then be retrieved using the TSNLP test suite categories, and linked via the semantics to particular generation system behaviour.

As a further example, consider variations of the following TSNLP items given on page 78 of (Estival et al., 1995).

He is given the book.
He is given the book by him.
The book is given to him.
The book is given to him by him.
* The book is given him.

The final item here is marked as ungrammatical although there are clearly contexts where (particularly with other tenses) it is acceptable. The Nigel grammar therefore generates all of these examples under particular modifications in the semantic input, as well as the additional orderings:

The book is given by him to him.
He is given by him the book.

Again, while these items include pragmatically marked orderings, there are specific contexts where they might be required. Here it would be particularly useful to employ the TSNLP notion of *interactions* between phenomena so that, for example, particular pronominalizations might be restricted in occurrence to appropriate word orders. In any event, all of these examples can also produce automatically marked up target suite items in the TSNLP-style. The second mapping of Table 1 applies and, for those cases where the indirect object precedes the direct object, the third mapping also.

Finally, we show an example of a non-toplevel target item, one of the nominal phrases of the list given in (Henschel, Bateman and Matthiessen, 1999): "An ebeniste whom the Marchands-Merciers employed most frequently". In TSNLP-terms, this is an example of the phenomenon NP_Modification-Relative_clauses-Restrictive, subtype Object-extraction. Here the result of automatic annotation (again given our mapping table above) is shown in Figure 5.[3]

```
An ebeniste
<UNIT
  class="CLAUSE"
  function="MOD-REL"
  features=
    "C_COMPLEMENTATION-DIVALENT-
DIRECT_OBJECT
     NP_MODIFICATION-RELATIVE_CLAUSES:
-RESTRICTIVE
        OBJECT-EXTRACTION"
  id=np-res-7>
  whom the Marchands-Merciers employed
most frequently
</UNIT>
```

Figure 5: Restrictive relative clause example

The semantics used as input specification for each of the example target items that we have illustrated here are also part of a complete specification. However, given the already noted diversity of these inputs across systems, this must be further differentiated so as to allow entries for different NLG systems, just as currently in the analysis-oriented TSNLP definitions the *outputs* are tagged.

[3]The TSNLP functor 'mod-rel' has been mapped from the Nigel grammar-specific grammatical function 'Event'.

Mapping	Nigel grammatical features	TSNLP categories
1	clause interrogative yes-no	S_Types-Questions-Y/N-questions-Inverted
2	clause passive-process recipiency	C_Diathesis-Passive-Divalent
3	clause operative complemented	C_Complementation-Divalent-Direct_Object
4	clause med-ben	C_Complementation-Trivalent-Indirect_object-Direct_object

Table 1: Examples of generator-specific to standardized category mappings

```
(e / (kick directed-action)
   :actor (m / man)
   :actee (b / ball))
```

"A man kicks a ball."

```
(e / (kick directed-action)
   :actor (m / man)
   :actee (b / ball)
   :pp-theme b))
```

"A ball is kicked by a man."

```
(e / generalized-possession
   :domain (m / man)
   :range (o / office))
```

"A man has an office."

```
(e / generalized-possession
   :domain (m / man)
   :range (o / office)
   :pp-theme o))
```

"A man has an office."

Figure 3: Contrasting behaviour depending on semantic and grammatical type

3. Extensions to annotation schemes motivated by generation

The use of standard categories such as those of the TSNLP suite illustrated here raises several issues and questions when the requirements of NLG are considered. As we have said, the notion of 'ill-formed' test items is particularly problematic. Providing such ill-formed input is seen as an essential part of the TSNLP design, but it is unclear what this should correspond to for generation. One example of this from TSNLP is the the category C_diathesis-middle; this identifies those constructions that look like transitive verbs but have no passive: i.e.,

'He has a house' vs. * 'A house is had'

The Nigel generation grammar naturally does not generate the latter ungrammatical example, but for this very reason it cannot then 'generate' the corresponding ill-formed item automatically.

There is a converse situation that *is* relevant for evaluating generation systems, however. And this is not generating ungrammatical sentences, but rather failing to generate sufficiently *differentiated* output given varying semantics. An example is shown in Figure 3 where generated strings are contrasted for two cases: in both cases the SPL semantic specification includes one input which contains the normal SPL idiom for creating passive constructions: `:pp-theme`. However the Nigel grammar fails (rightly) to produce the ungrammatical variant for the 'middle' case above. This means that it has also failed to find a grammatical way of expressing the requested semantics. This kind of differentiation should be reflected in a full target suite classification scheme for NLG evaluation.

The main omissions of current test suites as far as their suitability as candidate target suites is concerned is in the area of discourse. The DiEt project (Netter et al., 1998) has recognized this state of affairs in announcing its commitment to extending the range of phenomena covered by TSNLP to ellipsis and anaphora. This is good news for evaluators of NLG systems, but for designers of test data both of these phenomena imply the introduction of test items that are at least complex sentences, if not multi-sentential text spans. This would represent a radical departure from current practice.

The same implications follow from the need to ensure a generator's ability to explicitly and unambiguously signal rhetorical relations by the choice of appropriate discourse markers. Thus, to take just one illustrative example of a rhetorical relation–CONCESSION–we would want to test that at least the following wordings could be generated:

Although she felt tired, she went swimming.
She went swimming, although she felt tired.
She felt tired. Nevertheless, she went swimming.

Success would demonstrate a potential for flexibility which could be exploited according to thematic development, a heuristic which preferred to avoid repetition or another heuristic that selected the most specific marker available, for example.

A more ambitious goal would be to assess the ability of a generation system to make 'sensible' decisions concerning its output which are not strictly motivated by grammaticality. This has much to do with a system's ability to function robustly in the face of varying application contexts. The question of how to annotate or record this aspect of coverage is also open. For example, consider the following sentence (adapted from an example from (Zock, 1996)):

"When the old woman saw the little boy drowning in the river, she went to her canoe, in order to save him."

If we change the semantic specification for Nigel that generates this sentence so that the old woman becomes an old man–i.e., the *only* change in input is the gender of the person who sees the little boy, then Nigel generates:

”When the old man saw the little boy drowning in the river, the man went to his canoe, in order to save the boy .”

This is not a necessary choice; it is just being conservative in its pronominalization strategy. But since the goal of Nigel, as with many generators, is to produce a single best-shot at generation time, it has not taken the risk of producing an ambiguous result. Of course, if we then change the gender of the small person in the river too, then we get back to the dual situation for the first sentence generated, i.e.:

”When the old man saw the little girl drowning in the river, he went to his canoe, in order to save her.”

This kind of behaviour is relevant to application decisions and so it is motivated that a target suite should include annotations for such potentially ambiguous or problematic nominal referring expressions. This might constitute a further extension of the TSNLP notion of ‘interactions’. However, it is also the case that conducting this kind of evaluation requires a consensus on the input specification to the generator, and this is unlikely to come about in the near future.

4. Conclusions

In seeking to evaluate the grammatical coverage of the surface realization component of a natural language generation system, we considered the utility of re-using test suites designed to assess the coverage of natural language analysis systems. We found that they are of some interest, in helping inter-system comparisons and in providing an essential, application-oriented link to annotated corpora. But they have limitations. First, they contain a high proportion of ill-formed items which are inappropriate or problematic as targets for generation. Second, they omit discourse phenomena such as ellipsis, intersentential anaphora and markers of rhetorical relations which are key issues in producing texts that are concise yet unambiguous for their readers. We illustrated a partial remedy for this situation in the form of a text generator–KPML–that annotates its own output to an externally specified standard, the TSNLP scheme. We identified some open issues in the design of NLG target suites which are of equal interest to many members of the NLP community concerned with comprehension tasks, such as machine translation.

5. References

Bateman, John A., 1997. Enabling technology for multilingual natural language generation: the KPML development environment. *Journal of Natural Language Engineering*, 3(1):51–55.

Bateman, John A., 1999. *The KPML multilingual natural language generation system, development environment and tools*. At: http://purl.org/net/kpml

Coch, J. 1996. Evaluating and comparing three text production techniques. *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING 1996)*, Copenhagen, 249–254.

Elhadad, M. and Robin, J., 1996. A reusable comprehensive syntactic realization component. *Demonstrations and Posters of the 1996 International Workshop on Natural Language Generation (INLG ’96)*, 1–4.

Estival, D., et al., 1995. *The construction of test material*. TSNLP Deliverable D-WP3.1. At: http://tsnlp.dfki.uni-sb.de/tsnlp/.

Henschel, R., Bateman, John A. and Matthiessen, C. The solved part of NP generation. *Proceedings of the ESSLI’99 Workshop on the Generation of Nominal Expressions*.

Kasper, R., 1989 A flexible interface for linking applications to PENMAN’s sentence generator. *Proceedings of the DARPA Workshop on Speech and Natural Language*.

Lehmann, S. et al., 1996. TSNLP–test suites for natural language processing. *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING 1996)*, Copenhagen, 711–716.

Lavoie, B. and Rambow, O., 1997. A fast and portable realizer for text generation systems. *Proceedings of the Fifth. Conference on Applied Natural Language Processing*, 265–268.

Lester, J. C. and Porter, B. W, 1997. Developing and empirically evaluating robust explanation generators: the KNIGHT experiments. *Computational Linguistics*, 23(1):65–101.

Mellish, C. and Dale, R., 1998. Evaluation in the context of natural language generation. *Computer Speech and language*, 12:349–373.

Netter, K. et al., 1998. DiET–Diagnostic and evaluation tools for natural language processing applications. *Proceedings of the First International Conference on Language Resources and Evaluation*, Granada, pp. 573–579.

Oepen, S., Netter, K. and Klein, J., 1997. TSNLP–Test Suites for Natural Language Processing. In J. Nerbonne (ed.), *Linguistic databases*. Stanford, CA: CSLI Lecture Notes.

Sparck-Jones, K. and Galliers, J. R., 1996. *Evaluating natural language processing systems*. Berlin: Springer-Verlag.

van Noord, G. and Neumann, G., 1997. Syntactic generation. In R. Cole et al. (eds.), *Survey of the state of the art in human language technology*. Cambridge: Cambridge University Press, 147–150.

Zock, M., 1996 The power of words in message planning. *Proceedings of the 16th International Conference on Computational Linguistics*, 990–995.