Proceedings of the Corpus Linguistics 2005 Workshop on Using Corpora for Natural Language Generation

Anja Belz and Sebastian Varges (eds)

Contents

Preface

The Workshop on Using Corpora for Natural Language Generation was held in Birmingham on 14th July 2005, in conjunction with the Corpus Linguistics 2005 conference. The workshop was supported by the Association for Computational Linguistics Special Interest Group on Generation and received generous sponsorship from the British Academy. Intending to bring together researchers who use corpora for NLG research either in the traditional, manual way, or automatically, involving machine learning and statistical methods, the goal of the workshop was to present and discuss current research, to compare manual and automatic corpus exploitation, to evaluate achievements, and to identify challenges for the future.

The workshop's technical programme combined a keynote paper by Irene Langkilde-Geary, one of the founding members of the field of statistical NLG, with an introductory presentation by Roger Evans, eight regular papers, and two shorter presentations describing specific corpora and their possible uses for NLG. The programme was rounded off by a discussion on the topics of the workshop led by a panel whose members were Chris Brew, Irene Langkilde-Geary, Ehud Reiter and Bonnie Webber.

This volume presents the keynote paper by Irene Langkilde-Geary and the eight regular papers that were selected in a rigorous double-blind reviewing process from a substantial number of submissions from Australia, Brazil, England, Germany, Japan, Netherlands, Scotland and the United States.

We are pleased to say that there was a high level of interest in the workshop in general, with registrations from Australia, Denmark, England, Germany, Japan, Scotland and the United States. The workshop proceedings reveal that a lot of research effort continues to be devoted to the development and application of statistical and machine-learning methods for NLG. Such methods, and more generally methods for automatic corpus exploitation, may help to move the field of NLG towards the level of reusability and robustness achieved by NLU.

We would like to thank all the authors who submitted extended abstracts for the workshop, our programme committee who helped us put together an excellent workshop programme, the panel members and the organisers of the Corpus Linguistics 2005 conference, and everybody else who helped us take this workshop from an initial idea to a successful reality.

July 2005 Anja Belz and Sebastian Varges

Organisation

Workshop Co-chairs and Organisers

Anja Belz, ITRI, University of Brighton, UK
Sebastian Varges, CSLI, Stanford University, USA

Programme Committee

Anja Belz, ITRI, University of Brighton, UK
John Carroll, Informatics, University of Brighton, UK
Robert Dale, Macquarie University, Australia
Michel Genereux, ITRI, University of Brighton, UK
Kevin Knight, ISI, University of Southern California, USA
Chris Mellish, University of Aberdeen, UK
Stephan Oepen, Oslo University, Norway, and CSLI Stanford, USA
Sebastian Varges, CSLI, Stanford University, USA

Automatic Evaluation of Referring Expression Generation Using Corpora *

Surabhi Gupta and Amanda J. Stent
Computer Science Department
Stony Brook University
Stony Brook, NY 11794-4400 USA
sugupta@ic.sunysb.edu, stent@cs.sunysb.edu

Abstract

We report on a set of experiments using corpora to evaluate referring expression generation for spoken dialog. In dialog, participants frequently converge on the same referring expressions even if those referring expressions are inefficient. Existing rule-based algorithms for referring expression generation do not adequately model this adaptation. We extended two such algorithms with simple models of partner adaptation. We evaluated these algorithms automatically using corpora of spoken dialog.

Referring expression generation is an important issue because referring expressions are prevalent in all types of discourse, and mproper construction of a referring expression can result in referring expressions that are ambiguous (e.g. *the book* when there are two books) or that lead to false implicatures (e.g. *the bright red book* when there is only one book) [Grice, 1975]. The generation of referring expressions is uniquely dependent on pragmatic constraints [Jordan and Walker, 2005; Krahmer and Theune, 2000]. In particular, in dialog participants adapt to each other's choice of referring expressions [Brennan, 1996; Metzing and Brennan, 2003]. However, most existing algorithms for referring expression generation do not take into account partner-specific adaptation (c.f. [Jordan and Walker, 2005]).

In this paper, we report on a set of experiments using corpora to automatically evaluate algorithms for referring expression generation for dialog. We used the MapTask and Coconut corpora, which are described in Section 2. The algorithms we use are described in Section 3. We describe our evaluation in Section 4 and conclude with some ideas for future work in Section 5.

*We would like to thank the blind reviewers for their helpful comments. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010; and by the National Science Foundation under Grant No. 0325188.

1 Related Work

Referring expression generation is one of the most widely studied research problems in natural language generation. Dale and Reiter's Incremental Algorithm [Dale and Reiter, 1995] is the most widely used. There have been several extensions to the basic Incremental Algorithm, including extensions to increase the efficiency of the referring expressions generated [Gardent and others, 2003; Horacek, 2003] and to refine the contrast set based on consideration of the discourse context [Krahmer and Theune, 2000]. Other work on referring expression generation has expanded the types of expressions that can be generated, but usually at the cost of efficiency in generation (e.g. [Gardent and others, 2003; van Deemter, 2000; 2002; Varges and van Deemter, 2005]). For these approaches, no quantitative evaluation is reported.

Siddharthan and Copestake addressed the problem of open-domain noun phrase generation [Siddharthan and Copestake, 2004]. They evaluated their algorithm by computing how many generated noun phrases were identical to corresponding human-produced noun phrases from the Wall Street Journal, and reported a performance of 81.5%.

Other work has looked at statistical noun phrase generation from corpora Poesio et al. performed several experiments on learning to predict aspects of the realization of a noun phrase. They compared the answer provided by their learned models to annotated testing data. They report an accuracy of 70% for predicting noun phrase type (e.g. personal pronoun, bare NP), and of 67.5% for predicting the syntactic form of an attribute realization given semantic and pragmatic information [Cheng *et al.*, 2001; Poesio and others, 1999]. Shaw and Hatzivassiloglu [Shaw and Hatzivassiloglou, 1999] and Malouf [Malouf, 2000] trained several models for prediction of prenominal adjective ordering; performance varies, with best performance by corpus ranging from 71.04% on financial data to 94.9% on medical data. Roy used humans to evaluate output from his shape description generator in a forced-choice experiment and reported a performance of 81.3% [Roy, 2002].

Jordan and Walker trained a classifier on an annotated version of the Coconut corpus to select sets of attributes for inclusion in generated noun phrases based on pragmatic features and information from a knowledge base [Jordan and Walker, 2005]. Their goal was to examine which of three models of noun phrase generation (include Dale and Reiter's model) works best. Their best reported classification accu-

racy (combining all models) is 59.9%. They do not perform attribute ordering.

2 Data

For our comparison of approaches to referring expression generation we used two dialog corpora, the MapTask corpus [Anderson and others, 1991] and the Coconut corpus [Di Eugenio and others, 1998]. Table 1 shows a breakdown of referring expressions in our dialogs by type. We only used non-embedded definite and indefinite noun phrases in our experiments. Furthermore, we only used noun phrases that refer to some object represented in the world of the task. For example, for MapTask references to the paper on which the map is printed such as *the left hand side* are excluded from this study.

2.1 Corpora

Maptask In each MapTask dialog [Anderson and others, 1991], there is a giver and a follower. Each participant has a map with landmarks indicated; there are differences in the position and number of landmarks on the two maps. The giver's map has a route between the landmarks; the giver provides instructions to the follower so that the follower is able to reproduce the route on the giver's map.

Due to the labor involved in annotating the dialogs, we randomly selected 30 dialogs from the MapTask corpus to use for these experiments. We used 16 dialogs where the participants had eye contact and 14 where they did not.

Coconut In each Coconut dialog [Di Eugenio and others, 1998], each partner is given a partial inventory of furniture. Their goal is to buy as much furniture as possible for two rooms of the house, the living room and the dining room. They have to try and match colors of furniture within a room. The participants are given information about calculating points for the furniture that they have bought (e.g. a sofa is worth 200 points, dining table chairs are worth 50 points each). The dialog partners sat in different rooms and communicated with each other via text messaging.

What makes this corpus interesting is that there are multiple items of furniture with the same `type` attribute, which means more scope for modifier selection. We used all 16 unique dialogs in the Coconut corpus.

2.2 Dialog processing

For each selected dialog, we constructed an annotation file, lexicon and set of knowledge representations.

Annotation Files We processed each dialog by: extracting a plain-text transcript; part-of-speech tagging it using the Brill tagger [Brill, 1992], and automatically extracting noun phrases using a pattern matcher over the part-of-speech tag sequences. We then corrected the noun phrase extraction by hand, and annotated each extracted noun phrase for referent (in the knowledge representation), type (noun phrase or pronoun), and to indicate whether the noun phrase was embedded in another noun phrase. The result was a file containing all referring expressions for the dialog in order.

Knowledge Representations We constructed a knowledge representation for each speaker in each dialog by hand from examination of the materials provided to the speakers (maps, furniture inventories). The knowledge representations for MapTask included values for six attributes for each potential referent: contents (e.g. *sandstone* in *sandstone cliff*), size (e.g. *large*), state (e.g. *parched, disused*), location, color and type (e.g. *cliffs*). For Coconut, the knowledge representations included values for five attributes for each potential referent: quantity (e.g. *2* in *2 chairs*), cost (e.g. *300* in *the $300 chair*), state (e.g. *high* in *high table*), color and type (e.g. *chair*).

Lexicons We also constructed a lexicon by hand for each dialog. The lexicon contained a list of words that might appear in a noun phrase in that dialog. We labeled each word with its word sense number from WordNet, the category of the word in our knowledge representation (e.g. size, state, location) and the singular form of the word.

2.3 Choice of corpus

The choice of corpus is particularly important for evaluation of referring expression generation. First, the corpus should be rich in the referring expressions of interest. For example, Siddharthan and Copestake found only 146 definite descriptions in the Wall Street Journal that could be used in their evaluation [Siddharthan and Copestake, 2004]. Between our two corpora, there were significant differences in terms of NP type and number and location of modifiers, as shown in Table 1. In fact, we sacrificed a focus on spoken dialog in order to obtain richer noun phrases.

Second, it is important to be clear about which noun phrases are being studied and what proportion of the number of noun phrases in the corpus they represent, so that performance improvements for competing algorithms can be evaluated in terms of global impact. We used a modified version of the disclosure table described in [Byron, 2001].

Both of our corpora involve fairly straightforward referential tasks. By contrast, in corpora of descriptions collected by psycholinguists (e.g. [Brennan, 1996; Metzing and Brennan, 2003], the conversational partners make considerable use of their general world knowledge to assign new labels to objects. These data cannot be modeled by any existing algorithm for generating referring expressions.

3 Implementation

3.1 Algorithms

We implemented Dale and Reiter's Incremental Algorithm [Dale and Reiter, 1995], and Siddharthan and Copestake's recent extension to this algorithm [Siddharthan and Copestake, 2004]. We then implemented two approaches to modeling partner effects and three approaches to pre/postmodifier placement as extensions to each algorithm. We compare the performance of all of these algorithms to each other and to a simple baseline.

We performed our experiments using a program that works through the annotation file for each dialog, passing the referent in each line to the selected algorithm and comparing the output to the human-produced referring expression. Each algorithm takes as input a referent for which to generate a referring expression, a preference list of attributes to use in generating the referring expression and a contrast set of distractors for the referent constructed from the discourse context and

	NP category	MapTask	Coconut
Total Count			
	Def	2118	116
	Indef	1411	967
	1st person pro.	563	440
	2nd person pro.	1275	165
	3rd person pro.	614	79
Total NPs		5981	1767
Excluded			
	Quantity Nouns	160	291
	Pron (first+second+third)	2452	684
	NPs not in KR, Other	2075	321
Included			
	No Modifiers	113	13
	Simple NPs	1268	229
	Complex NPs	26	242
NPs used		1294	471
Mean # attributes		2.02	3.07

Table 1: Comparison of Noun phrases in corpora used

from the knowledge representation for the current speaker. The algorithms all output an ordered list of attribute values that approximate the final referring expression.

The referent is taken from the annotation file for each dialog. The preference list is an ordered list of the attributes in the knowledge representation for that dialog. The ordering of the preference list varies by algorithm.

We look up each referent in our model of discourse context. If it is there, then the initial contrast set is the entities in the discourse context. If it is not, then the initial contrast set is the entities in the discourse context plus the entities in the current speaker's knowledge representation. After generating a referring expression, we add its referent to the discourse context. This presumes that each referring expression is perfectly understood and ignores the important role played by grounding in conversation [Clark, 1996; Cahn and Brennan, 1999]. However, in this research we did not attempt to interpret all utterances in the discourse, so this is a necessary approximation.

Baseline The initial referring expression consists solely of the value of the `type` attribute. The remaining attributes are selected at random from those attributes for which the referent has a value, until all distractors from the contrast set have been eliminated. The value of each selected attribute is prepended to the referring expression. This algorithm is equivalent to the Dale and Reiter algorithm with no ordered preference list.

Dale and Reiter Dale and Reiter's Incremental Algorithm [DR] was designed to model cognitive efficiency constraints that affect how humans produce referring expressions. The key difference from the baseline algorithm is that a preference list (an ordered list of attributes prespecified by the system designer) is used to select the next attribute for inclusion in the referring expression. As in the baseline algorithm, the value for each selected attribute is prepended to the referring expression. For the MapTask corpus, we used the following preference list: <`type, size, content, state, location, color`>. For the Coconut corpus, we used this preference list: <`type, color, cost, quantity, state`>.

Siddharthan and Copestake Siddharthan and Copestake's extension to the Incremental Algorithm [SC] was designed to permit it to be used for domains for which there do not exist detailed ontologies, especially open domain noun phrase generation. The basic algorithm is identical, except for the ordering of the preference list. For each referring expression, the attributes in the preference list are reordered so as to prefer maximally distinctive adjectives (attribute values).

Partner-Specific Adaptation In order to model partner effects, we simply consider previous mentions of the current referent. In our basic approach [DRP, SCP], we reorder the preference list for each referent to reflect the ordering of attributes in the most recent mention of that referent (by either speaker), with unmentioned attributes coming at the end of the preference list.

Partner-Specific Adaptation Variant In a variant of our basic approach to modeling partner effects [DRPV, SCPV], we not only reorder the preference list to reflect those attributes used in the most recent mention of the current referent, but also *require* that attributes used in the most recent mention be used in the referring expression to be generated, even if those attributes do not eliminate any distractors.

3.2 Generating Postmodifiers

The Incremental Algorithm only selects and orders attributes that will be used to generate noun phrase premodifiers. As Table 1 shows, noun phrases containing postmodifiers accounted for approximately 51% of the indefinite and definite noun phrases in the Coconut corpus. We implemented three extensions to this algorithm that permit the generation of complex noun phrases including postmodifiers as well as premodifiers. These extensions were all evaluated in combination with each of the six algorithms described in the previous section. For each dialog in each corpus, we used the other dialogs in that corpus to compute probabilities for these models.

Random In this approach, the decision about whether to place the attribute value before or after the head noun is made at random. The ordering of premodifiers is still right to left as specified in the preference list; the ordering of postmodifiers is left to right as specified in the preference list.

Unigram In this approach, the decision about whether to place the attribute value before or after the head noun is made by considering the relative frequency of placement of this attribute in the corpus under consideration. For example, for each of the sixteen dialogs in the Coconut corpus, we compute the relative frequencies of occurrence of each attribute before and after the head nouns in in the noun phrases of the other fifteen dialogs. During generation, we decide whether to construct a premodifier or postmodifier from a particular attribute value based on whether the relative frequency of occurrence of the attribute in premodifiers was greater than or less than the relative frequency of occurrence of the attribute in postmodifiers. The ordering of premodifiers is still right to left as specified in the preference list; the ordering of postmodifiers is left to right as specified in the preference list.

Bigram This approach also relies on knowing the relative frequencies of occurrence of the attributes in the corpus. However, in this case we compute relative frequencies in context: for each attribute, we compute the probability of occurrence of that attribute given each other attribute. During generation, we look up the relative frequencies of the current attribute occurring before and after each attribute already in the referring expression, and place the attribute value at the location of highest relative frequency. Note that this means that modifiers are not necessarily ordered as specified in the preference list; the preference list determines which modifiers are used, but the corpus frequencies determine where they are placed (cf. [Shaw and Hatzivassiloglou, 1999; Malouf, 2000]).

4 Experiments

In these experiments, we were interested in approaches to modeling partner-specific adaptation, so could not evaluate solely for comprehensibility (cf. [Roy, 2002]). Our algorithms perform attribute selection and ordering, so we could not use classification accuracy as in [Jordan and Walker, 2005]. At the same time, we were not performing surface realization, so could not use string equality (cf. [Siddharthan and Copestake, 2004]).

Our assumption was that humans are the best generators of referring expressions in conversation. The performance of a program modeling partner-specific adaptation can therefore be automatically evaluated by comparison with human-produced output. Of course, this assumption sometimes fails. For example, a human-produced referring expression may not be understood by the conversational partner; or it may be non-optimal (too long, or using attribute values that are not clear to the conversational partner). However, it is not feasible to interrupt a dialog at every referring expression to query the hearer as to whether it is "optimal".

Another issue is that two referring expressions may be equally good in context; our evaluation method does not take this into account. However, it evaluates the phenomena we are interested in (attribute selection, attribute ordering and partner-specific adaptation) and can be performed automatically, even without the use of an end-to-end dialog system. It is therefore a reasonable compromise for efficient evaluation and comparison for this task.

To evaluate our algorithms, the output referring expression (consisting of a sequence of attribute values, possibly including a determiner and almost always including `type`) is compared to the human-produced referring expression from the annotation file[1]. We automatically compute:

- (C) the number of attribute values that are correct
- (I) the number of attributes that are inserted
- (D) the number of attributes that are deleted
- (M) the number of attributes that are moved

Because we used domain-specific lexicons, we never had an incorrect value for an attribute. Our score for each generated referring expression is S = C / C + I + D + M; this score ranges from 0 to 1, with higher numbers indicating referring expressions that better match the original human-produced ones. For evaluating our algorithms, we computed the mean score over all noun phrases for each corpus.

4.1 Hypotheses

We compared the six algorithms described earlier, and our baseline algorithm, in combination with our three extensions for pre/postmodifier placement. Tables 2 and 3 show the mean scores and number of items scoring 1 for each algorithm for each corpus. We used paired t-tests to compare the performance of our algorithms. Our hypotheses were as follows:

1. DR and SC will outperform baseline
2. SC will outpeform DR
3. DRP and DRPV will outperform DR
4. SCP and SCPV will outperform SC
5. For each version of the Incremental Algorithm, Bigram will outperform Unigram which will outperform Random

There were differences in our two corpora that led to slightly different results for each one. In particular:

- For MapTask our knowledge representations were incomplete (we did not have attribute values for all attributes for all referents)
- For Coconut there were more distractors on average
- For each MapTask dialog, the two participants' knowledge representations had conflicts; for Coconut, they were complementary

MapTask

Our basic results for MapTask, based on mean scores, are:

1. DR and SC perform significantly *worse* than baseline for except for Random ($p < 0.01$).

[1] A small number were excluded at this stage because the original NP was incomplete: 20 for Maptask and 7 for Coconut.

Mean	None	Random	Unigram	Bigram
Baseline	0.636 (223)	0.622 (220)	0.636 (223)	0.636 (223)
DR	0.633 (216)	0.625 (216)	0.633 (216)	0.633 (216)
DRP	0.638 (232)	0.621 (222)	0.638 (234)	0.638 (234)
DRPV	0.764 (639)	0.657 (409)	0.760 (630)	0.764 (638)
SC	0.631 (232)	0.618 (223)	0.630 (232)	0.630 (232)
SCP	0.631 (232)	0.618 (225)	0.630 (232)	0.630 (232)
SCPV	0.760 (636)	0.664 (436)	0.755 (627)	0.762 (635)

Table 2: Mean scores and number perfect for MapTask

2. SC does *not* outperform DR. Both the algorithms give similar results except for Random, where SC performs significantly *worse* than DR ($p < 0.01$).
3. DRP performs significantly better than DR except for Random ($p < 0.01$). DRPV performs significantly better than DR for all postmodifier variants ($p < 0.01$).
4. SCP does *not* perform better than SC. However, SCPV does perform significantly better than SC for all postmodifier variants ($p < 0.01$).
5. Both Bigram and Unigram outperform Random ($p < 0.01$). Bigram outperforms Unigram for DRPV and SCPV. ($p < 0.01$).

DR does not perform significantly better than baseline because after selecting the `type` attribute, there are no remaining distractors for most of the referents in this corpus. However, the algorithms that model partner effects typically do outperform those that do not because most of the noun phrases do include at least one attribute other than type; in other words, the human-produced noun phrases are verbose.

Coconut

Our basic results for Coconut are:

1. DR and SC do perform significantly better than baseline for all postmodifier variants ($p < 0.01$).
2. SC performs significantly better than DR for all postmodifier variants ($p < 0.05$).
3. DRP and DRPV both perform significantly better than DR for all postmodifier variants ($p < 0.01$).
4. SCP does *not* perform better than SC except for Random ($p < 0.05$). SCPV performs significantly better than SC for all postmodifier variants ($p < 0.01$).
5. There is no significant difference between Random and Unigram. Bigram performs better than Random for SC, SCP and SCPV ($p < 0.05$). Bigram performs significantly better than Unigram for SC, DRP, SCP, SCPV and DRPV ($p < 0.05$).

Because there are more distractors on average for a referent in the Coconut corpus, it is less likely that simply selecting the type attribute will remove all the distractors. So, for this corpus, DR and SC perform better than baseline. Because there are more attributes per referent, there are more ways to incorrectly place an attribute value in the NP, which explains why Random and Unigram are more similar for this corpus.

For both corpora, SCP does not perform significantly better than SC, but SCPV does. We think that with SCP, the reordering of attributes leads to "over-efficient" NP generation compared to SCPV. This does not happen with DRP, for which the preference list is not re-ordered on a per-referent basis.

Discussion

Prior work in cognitive psychology has shown that humans frequently produce "non-optimal" referring expressions, and has highlighted the need to model partner effects in interpreting and producing referring expressions [Brennan, 1996; Metzing and Brennan, 2003]. The results reported here provide confirmation that our relatively simple extensions to the Incremental Algorithm can provide effective modeling of partner effects in referring expression generation, compared to state-of-the-art statistical approaches [Jordan and Walker, 2005]. These results also show that our extension to the Incremental Algorithm to permit generation of NP postmodifiers does in fact improve the algorithm's performance across both corpora.

Finally, these results show that it is important to consider the domain when selecting an algorithm for referring expression generation. When there is only one object of a particular type, our baseline algorithm performs remarkably well. While there are some important domains with multiple distractors for each possible referent (e.g. air travel, rail travel, purchasing domains in general), there are many that do not have this characteristic (e.g. customer service, tutoring, some planning domains).

There are some types of referring expression that our algorithms could not handle. First, sometimes a conversational partner seems to have invented information about an object. For example, in the MapTask corpus on some maps there is a location called "highest viewpoint", which is drawn on the map with birds flying around it. These birds were referred to as seagulls. Second, we do not generate descriptions of sets (cf. [van Deemter, 2002]). In the Coconut corpus in particular there were many complex set descriptions such as *2 green and 2 red chairs both for $100 each*. Third, we do not generate pronominal references. Adding the capability to generate third-person pronouns would require only simple modifications to our existing algorithms to track discourse focus. We know of no generation algorithm that generates first and second person pronouns; pronouns such as *we* and *they* are references to sets and so are particularly problematic.

5 Conclusions and Future Work

In this paper we analyze different approaches to generating referring expressions for spoken dialog. We show that with

Mean	None	Random	Unigram	Bigram
Baseline	0.611 (72)	0.609 (72)	0.610 (72)	0.610 (72)
DR	0.637 (74)	0.638 (74)	0.637 (74)	0.637 (74)
DRP	0.674 (112)	0.684 (115)	0.683 (115)	0.685 (115)
DRPV	0.760 (165)	0.763 (165)	0.761 (165)	0.764 (165)
SC	0.657 (114)	0.655 (114)	0.656 (114)	0.659 (114)
SCP	0.660 (121)	0.661 (121)	0.660 (121)	0.663 (121)
SCPV	0.753 (163)	0.754 (163)	0.754 (163)	0.757 (163)

Table 3: Mean scores and number perfect for Coconut

relatively simple approaches to modeling partner-specific adaptation, it is possible to improve the performance of algorithms for referring expression generation. These algorithms can also be improved by simple extensions that permit the generation of noun phrases involving postmodifiers.

We plan to integrate our implemented algorithms into a spoken dialog system for collaborative problem solving being developed partly at Stony Brook University. This will give us an opportunity to evaluate the performance of our algorithm in-use, as opposed to off-line.

References

[Anderson and others, 1991] A. Anderson et al. The HCRC Map Task corpus. *Language and Speech*, 34, 1991.

[Brennan, 1996] S. E. Brennan. Lexical entrainment in spontaneous dialog. In *Proceedings of ISSD 1996*, 1996.

[Brill, 1992] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of ANLP 1992*, 1992.

[Byron, 2001] D. Byron. The uncommon denominator. *Computational Linguistics*, 27(4), December 2001.

[Cahn and Brennan, 1999] J. Cahn and S. E. Brennan. A psychological model of grounding and repair in dialog. In *Proceedings of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, 1999.

[Cheng *et al.*, 2001] H. Cheng, M. Poesio, R. Henschel, and C. Mellish. Corpus-based NP modifier generation. In *Proceedings of NAACL 2001*, 2001.

[Clark, 1996] H. Clark. *Using Language*. Cambridge University Press, 1996.

[Dale and Reiter, 1995] R. Dale and E. Reiter. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2), 1995.

[Di Eugenio and others, 1998] B. Di Eugenio et al. An empirical investigation of proposals in collaborative dialogues. In *Proceedings of COLING-ACL 1998*, 1998.

[Gardent and others, 2003] C. Gardent et al. Generating definite descriptions: Non-incrementality, inference, and data. In T. Pechman and C. Habel, editors, *Multidisciplinary approaches to language production*. Walter de Gruyter, Berlin, 2003.

[Grice, 1975] H. P. Grice. Logic and conversation. In *Syntax and Semantics: Vol 3, Speech Acts*. Academic Press., New York, 1975.

[Horacek, 2003] H. Horacek. A best-first search algorithm for generating referring expressions. In *Proceedings of ACL 2003*, 2003.

[Jordan and Walker, 2005] P. Jordan and M. Walker. Learning content selection rules for generating object descriptions in dialogue. *Journal of Artificial Intelligence Research*, 4, 2005.

[Krahmer and Theune, 2000] E. Krahmer and M. Theune. 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, 2000.

[Malouf, 2000] R. Malouf. The order of prenominal adjectives in natural language generation. In *Proceedings of ACL 2000*, 2000.

[Metzing and Brennan, 2003] C. Metzing and S.E. Brennan. When conceptual pacts are broken. *Journal of Memory and Language*, 49, 2003.

[Poesio and others, 1999] M. Poesio et al. Statistical NP generation: A first report. In *Proceedings of the ESSLLI Workshop on NP Generation*, 1999.

[Roy, 2002] D. Roy. Learning visually grounded words and syntax for a scene description task. *Computer Speech and Language special issue on Spoken Language Generation*, 16(3–4), 2002.

[Shaw and Hatzivassiloglou, 1999] J. Shaw and V. Hatzivassiloglou. Ordering among premodifiers. In *Proceedings of ACL 1999*, 1999.

[Siddharthan and Copestake, 2004] A. Siddharthan and A. Copestake. Generating referring expressions in open domains. In *Proceedings of ACL 2004*, 2004.

[van Deemter, 2000] K. van Deemter. Generating vague descriptions. In *Proceedings of INLG 2000*, 2000.

[van Deemter, 2002] K. van Deemter. Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1), March 2002.

[Varges and van Deemter, 2005] S. Varges and K. van Deemter. Generating referring expressions containing quantifiers. In *Proceedings of IWCS-6*, 2005.

Re-Creating Dialogues from a Corpus

Amy Isard and **Carsten Brockmann** and **Jon Oberlander**
HCRC, University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, UK
{Amy.Isard,Carsten.Brockmann,J.Oberlander}@ed.ac.uk

Abstract

We describe the collection and annotation of a corpus of dialogues about movies, and a system which uses utterances from this corpus in generating dialogues which vary according to the personalities assigned to two characters.

1 Introduction

This paper describes the design and implementation of the first version of the Critical Agent Dialogue (CrAg) system, which generates dialogues which vary according to the personalities assigned to two characters, based on recent research into different vocabulary, syntax and dialogue strategies exhibited according to personality type [Gill and Oberlander, 2002]. This research uses Eysenck's three factor model [Eysenck and Eysenck, 1991], in which personality is described in terms of the three dimensions psychoticism, extraversion, and neuroticism, each of which can separately influence language production.

This is not a dialogue system in the usual sense; both sides of the conversation are generated, in order to be able to manipulate the interactions between participants with different personalities. The output is currently being evaluated by human subjects, to ascertain whether they can recognise the personality characteristics which we have mimicked.

This system has some similarities to the NECA system [Piwek, 2003], but rather than focussing on the discourse structure, we have concentrated on the component of the system which assigns personality scores to utterances.

The dialogues are constructed using utterances from a corpus (see Section 2) which are ranked and combined into a coherent dialogue using the techniques described in Sections 3 and 4. Some example dialogues are presented in Section 4.4.

2 Corpus

2.1 Collection

Ten pairs of participants went to see a film of our choosing and were later recorded having a conversation about it. Three films were chosen which were showing at the same time, and were from three different genres: "League of Extraordinary Gentlemen" (action, sci-fi, fantasy), "Mystic River" (drama, crime) and "Intolerable Cruelty" (romantic comedy). The dialogues were recorded in a soundproof room, and participants were told that they could talk about any aspect of the film of their choosing, and asked to try to stay on the topic of the film they had just seen, but the conversation was not monitored. Dialogues ranged in length from 12 to 25 minutes, with an average of 19 minutes.

2.2 Transcription and Annotation

The dialogues were segmented into phrases and transcribed orthographically using the Transcriber tool [Barras *et al.*, 2001]. The dialogues were then annotated using the NITE XML Toolkit (NXT [Carletta *et al.*, 2003]). This toolkit provides utilities which allow users to create their own annotation interface. An interface was created which displayed the two participants' speech in separate windows, and allowed the annotator to listen to the speech and to combine phrases into utterances, while annotating the utterances as described below. A screenshot of the tool can be seen in Figure 1.

Topics

The annotator assigned one or more topics to each utterance from a pre-defined list, shown below. Topics without definitions are assumed to be self-explanatory.

- action sequences
- actors
- characters
- cinematography style: the look of the film
- dialogue
- directing: directing style, director(s)' intentions etc.
- humour
- music
- romance
- special effects
- whole movie
- other this film: a topic related to this film not included in the above list
- other film: a discussion about another film or films
- not film related: any discussion not related to films at all

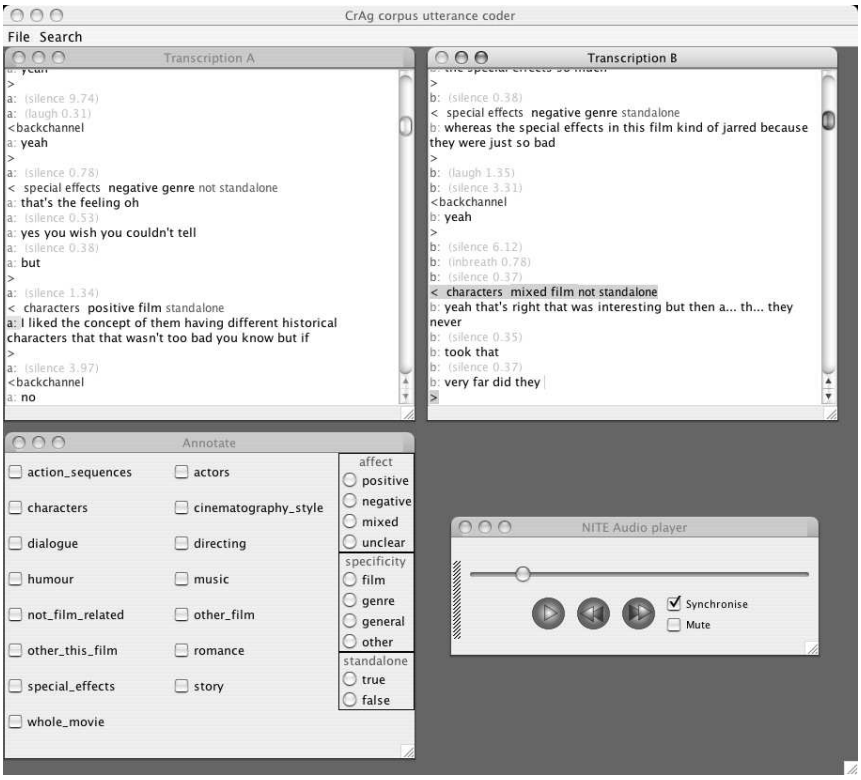

Figure 1: NXT topic annotation tool

- none: utterances where topic cannot be assigned, e.g. "um", "he it" "I ... I think'

Affect

The annotator also chose the affect of the utterance from the following list (one per utterance)

- positive
- negative
- mixed – both positive and negative e.g "all the cinematography was alright there was nothing interesting in it nothing daring"
- unclear
  - neutral e.g. "what did you think of Sean Connery", "well there's a clear implication that they had a relationship before"
  - not possible to assign affect – unclear is automatically assigned to all utterances whose topic is "none"

Generality

With re-generation in mind, utterances were labelled according to whether they make sense out of context. This means that most utterances with anaphoric references are rejected e.g. "there was no mention of that at all in the film" along with those which require knowledge of the previous utterance e.g. "and there wasn't even that much blood-sucking which is kind of disappointing for a vampire" and questions e.g. "what did you think?"

For the same reason, the utterances were also ranked for whether they could apply to just one film (e.g. "they'd start little storylines like when Sean Connery was teaching the American chap to shoot") or could be used to discuss any film (e.g. "I don't have anything positive to say about it actually").

2.3 Corpus Statistics

Topics/Films	LXG	IC	MR	all
action sequences	11	0	16	27
actors	30	66	95	171
characters	110	52	282	444
cinematography style	7	0	12	19
dialogue	37	8	8	53
directing	23	48	65	136
humour	5	76	2	83
music	0	0	25	25
romance	0	9	8	17
special effects	48	0	0	48
story	165	83	245	493
whole movie	74	36	44	154
other	106	173	124	403
Total	427	401	637	1465

Table 1: All Utterances

This resulted in a total of 1465 utterances averaging 73 per speaker. The topics are not distributed evenly throughout the dialogues since we used films from three different genres, and

some topics (e.g. special effects) do not apply to all types of film.

Table 1 includes all the utterances in the corpus (N.B. because there can be more than one topic per utterance, the totals at the bottom are less than the sum of their columns).

Table 2 shows all the utterances which were considered to be usable for re-generation. Utterances listed under each film are those which could only be used in a discussion of that particular film, and those in the column "general" could be used to talk about any film (see Section 2.2).

Topics/Films	LXG	IC	MR	general	all
action sequences	6	0	1	0	7
actors	1	4	12	2	19
characters	7	1	12	3	23
cinematography style	5	0	2	2	9
dialogue	3	2	0	4	9
directing	1	3	2	1	7
humour	2	8	0	0	10
music	0	0	1	0	1
romance	0	0	0	0	0
special effects	14	0	0	0	14
story	14	2	12	7	35
whole movie	10	4	11	19	44
Total	44	15	41	32	132

Table 2: Re-usable Utterances

3 Ranking Utterances by Personality Features

3.1 Framework

The Critical Agent Dialogue (CrAg) 1.0 system is implemented as a collection of Open Agent Architecture (OAA, [Cheyer and Martin, 2001]) agents. Each agent is a program designed to fulfil a specific task; it informs the special OAA facilitator agent about its capabilities. Whenever an agent requires a task to be resolved, it sends a request to the facilitator, which then invokes the agent that can deal with the request, and returns the results to the requesting agent.

3.2 Augmenting the Annotation

In a first stage, the corpus utterances' annotation is augmented with information from a variety of linguistic and psycholinguistic resources. This knowledge is then used to compute neuroticism and extraversion scores (see Section 3.3).

Part of Speech Tagging and Lemmatisation

Each utterance is split into sentences, tokenised, and tagged with part of speech information using the *mxpost* part of speech tagger [Ratnaparkhi, 1996]. The *morph* tool [Minnen *et al.*, 2001] then determines each word's lemma form.

Based on the lemmata, we compute each utterance's type/token ratio, which measures the variety of words used; it equals 1 if every type is used only once, and decreases with each repetition.

MRC Psycholinguistic Database

The annotation is further augmented by information from the MRC Psycholinguistic Database (MRC PDb, [Wilson, 1988]), a machine readable dictionary of 150,837 words. For each word, it specifies up to 26 linguistic and psycholinguistic attributes, e.g.:

- written/spoken word frequencies
- familiarity, concreteness, imageability
- meaningfulness
- age of acquisition
- part of speech
- phonetic transcription, stress pattern

Linguistic Inquiry and Word Count

Linguistic Inquiry and Word Count (LIWC2001, [Pennebaker *et al.*, 2001]) is another machine readable dictionary. 2,300 words and word stems are annotated with one or more of 74 categories, e.g.:

- linguistic dimensions (pronouns, negations, articles, ...)
- psychological processes
  - positive/negative emotions
  - cognitive processes (insight, certainty, ...)
  - perceptual processes (seeing, hearing, feeling)
  - social processes (friends, family, ...)
- relativity (time, space, motion)
- personal concerns (occupation, leisure, physical states, ...)

The Formality Measure *F*

From each utterance's part of speech annotation we compute the formality measure F [Heylighen and Dewaele, 2002]; the authors propose the concept of formality as a "dimension of variation between linguistic expressions". The measure is based on frequency percentages of different word classes:

$$\begin{aligned} F = &(\text{noun freq.} + \text{adjective freq.} + \text{preposition freq.} \\ &+ \text{article freq.} - \text{pronoun freq.} - \text{verb freq.} \\ &- \text{adverb freq.} - \text{interjection freq.} + 100)/2 \end{aligned} \quad (1)$$

In Heylighen and Dewaele's study, oral female ($F = 38.7$) and oral male ($F = 41.6$) language was classified as informal; novels ($F = 52.5$) were average, while scientific text ($F = 65.7$) and newspapers ($F = 68.1$) ranked high on the formality scale.

3.3 Feature Combination

Previous research identified features characteristic for the language of extravert or neurotic speakers [Gill and Oberlander, 2002; Oberlander and Gill, 2004]. According to these results, we combine the utterance scores computed during the annotation phase using additive multiattribute value functions (AMVF). AMVF have been applied to represent user preferences [Carenini and Moore, 2000]; we use an implementation done for the user modelling component of the FLIGHTS system [Moore *et al.*, 2004].

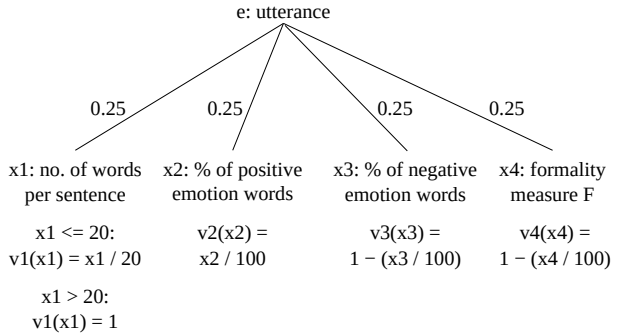

Figure 2: Partial additive multiattribute value function (AMVF) for extravert language.

In an AMVF, a value tree specifies the hierarchy of aspects of an entity e. Edges are weighted (w) according to the importance of their contribution to the parent node. For each leaf, a component value function v_i maps attribute value x_i to the $[0,1]$ interval (1 is most preferable). The weight w_i of a leaf node is computed as the product of the weights from the tree's root down to the leaf. Given this model, the value $v(e)$ of entity e can be computed:

$$v(e) = v(x_1, \ldots, x_n) = \sum w_i v_i(x_i) \quad (2)$$

A simplified example AMVF for extravert utterances is shown in Figure 2. Our complete set of features characteristic for high extravert language is listed below:

- high:
  - number of words per sentence
  - number of sentences per utterance
  - percentage of conjunctions (part of speech tag)
  - mean frequency count of spoken English (MRC PDb)
  - percentage of certainty words (*always*, *never*; LIWC)
  - percentage of positive emotion words (*happy*, *pretty*, *good*; LIWC)
  - percentage of social process words (*talk*, *us*, *friend*; LIWC)
- low:
  - percentage of determiners (part of speech tag)
  - mean concreteness (MRC PDb)
  - percentage of negation words (*no*, *never*, *not*; LIWC)
  - percentage of negative emotion words (*hate*, *worthless*, *enemy*; LIWC)
  - formality (F measure, cf. Section 3.2)
  - lemma-based type/token ratio

The following features characterise high neurotic language:

- high:
  - percentage of first person singular words (*I*, *my*, *me*; LIWC)
  - percentage of negative emotion words (LIWC)
- low:
  - percentage of determiners (part of speech tag)
  - percentage of positive emotion words (LIWC)
  - formality (F measure)

In the current version of the system, all features are given equal weight; we plan to fine-tune the weight adjustments in future.

4 Re-Generating Dialogue

4.1 Initialisation

Computer characters are defined by values for the personality dimensions extraversion (E), neuroticism (N), and psychoticism (P). These values are given in a range from 0 (low) to 1 (high). For psychoticism, in our current implementation, only the two settings low ($P < 0.5$) and high ($P \geq 0.5$) are distinguished, as explained below. The characters are also each assigned an agenda of topics about which they would like to talk; for each topic, their opinion about it (the *polarity*) is either positive or negative.

Dialogues between two computer characters are then regenerated by the OAA CrAg Steering Agent. Two character definitions and one of the three available films are selected, and the number of turns to generate is set.

4.2 The Affective Language Production Model

The generation process is informed by the Affective Language Production (ALP) model, developed by Oberlander and Gill. The simplest version of this model (ALP-1) starts from the idea that high extraverts have plenty of resource for linguistic interaction, but need to put less of it into detailed planning. High neurotics have less resource for linguistic interaction in the first place. It follows that extraversion finds its effects mostly at the stages of formulation (surface realisation). That is, the process and representations used in realisation differ between high and low extraverts. Secondly, neuroticism finds its effects at the stage of conceptualisation (content selection). That is, the process and representations used in content selection differ between high and low neurotics. Since content selection precedes surface realisation, variations in neuroticism will have consequences beyond the content selection stage, but this is their primary locus.

ALP-2 adds to ALP-1 a simple treatment of psychoticism. The idea is that some individuals pay more attention to their partner's utterances than others. Those who pay attention are more likely to produce utterances which overlap in both content (and form) with those of their partner.

4.3 Re-Generation

Thus, the re-generation process operationalises ALP-2 in the following way.

First, the character with the higher extraversion score begins the dialogue, and the topic at the top of their agenda is used.

Subsequent topic choice is influenced by the characters' psychoticism score. A low psychotic character continues on

the same topic as the previous speaker, while a high psychotic changes topic to the one at the top of their own agenda.

Next the corpus is queried for all utterances with the appropriate film, topic, and polarity. By the previous annotation process (cf. Section 3.2) each utterance was assigned an N and an E score.

A character's N level relates to content selection. Thus, the N score is the first filter we apply to rank the utterance candidates. To implement this as simply as possible, we keep the half of the candidates which most closely match the desired N score, and discard the other half.

A character's E level relates to surface realisation behaviour. So the remaining candidates are sorted by E score, and we choose the one that is closest to the character's desired E score as their actual utterance.

After this, it is the other character's turn to say something. The routine loops back to the point of topic choice.

4.4 Example Dialogues

Here we present three example dialogues with different combinations of character personalities. In all these examples, character **a** begins the dialogue, and in the first two cases, since **a** is HiP and **b** is LoP, we end up with two utterances on each topic, as **b** always aligns to **a**'s topic, and **a** always starts a new subject. In the third dialogue, both characters are LoP, so they will continue speaking on the same topic until they run out of things to say.

Dialogue 1: Mystic River

Character a HiE/MidN/HiP

Character b LoE/MidN/LoP

a (characters) I mean I think maybe maybe one of the things is it had too many characters and you know you can't do a character building thing with you know three main characters two cops and you know all the supporting family and all that

b (characters) the other thing the eventual murderer the kid brother I don't think his character was very well developed

a (whole movie) you know what it was like for me it was like a b grade book that's been made into a movie

b (whole movie) for a mystery crime it's absolutely crap

a (story) I didn't really care for the story too much you know there are certain films that have good murder mysteries and you know the person dies at the beginning but you still don't know who did it or what happened but this didn't really engage I guess

b (story) it could have been just that there's a a big grand vision of like all these lives intertwined that was cut down

Dialogue 2: League of Extraordinary Gentlemen

Character a LoE/MidN/HiP

Character b LoE/MidN/LoP

a (characters) I think the particular disappointment for me was was not getting the characters interacting and kind of jostling enough

b (characters) I mean there is this mix of different well known characters but they never sort of explore their personalities in any way

a (special effects) maybe it was an incredibly low budget film and so they had to do everything on computers

b (special effects) the the set where the city scenes and buildings they were just so obviously fake as to be unreal I mean

a (whole movie) it was a bit dull

b (whole movie) I was on the whole disappointed with it

Dialogue 3: League of Extraordinary Gentlemen

Character a MidE/MidN/LoP

Character b MidE/MidN/LoP

a (whole movie) so basically we have nothing good to say about it

b (whole movie) I have to say this is very rare this is by far the the worst film I've seen in a in a very long time

a (whole movie) they didn't do anything particularly well at all

b (whole movie) I've seen a lot of kind of bad science fiction adventure fantasy type films before but this one is sort of bad in ways that none of the other bad films are bad you know

a (whole movie) I think probably I don't know I really thought it was going to be better than it was I think that the big names carried it and if you just took big names out of that movie it wouldn't have any I think it's a tv movie

b (whole movie) I'm still trying to grab something that I liked about it

5 Future Work

The first version of our generation system selects utterances from a corpus of human dialogues about films. The next version, currently under development, will use the OpenCCG realiser [White and Baldridge, 2003] to create more varied and flexible dialogues on the same subject, using the personality ranking algorithms presented here. We will also introduce the use of alignment as described in [Brockmann *et al.*, 2005].

There are many other possible applications for the ranking method presented in Section 3. It could for example be used in a real-time dialogue between a user and a system, to evaluate the probable personality of the user and provide appropriate responses.

References

[Barras *et al.*, 2001] Claude Barras, Edouard Geoffrois, Zhibiao Wu, and Mark Liberman. Transcriber: development and use of a tool for assisting speech corpora production. *Speech Communication*, 33(1–2):5–22, 2001. See also `www.etca.fr/CTA/gip/Projets/Transcriber`.

[Brockmann *et al.*, 2005] Carsten Brockmann, Amy Isard, Jon Oberlander, and Michael White. Modelling alignment for affective dialogue. In *Proceedings of the Workshop on Adapting the Interaction Style to Affective Factors*, Edinburgh, July 2005. To appear.

[Carenini and Moore, 2000] Giuseppe Carenini and Johanna D. Moore. A strategy for generating evaluative arguments. In *Proceedings of the 1st International Natural Language Generation Conference (INLG-00)*, pages 47–54, Mitzpe Ramon, Israel, 2000.

[Carletta *et al.*, 2003] Jean Carletta, S. Evert, U. Heid, J. Kilgour, J. Robertson, and H. Voorman. The NITE XML toolkit: flexible annotation for multi-modal language data. *Behavior Research Methods, Instruments, and Computers*, 35(3):353–363, 2003.

[Cheyer and Martin, 2001] Adam Cheyer and David Martin. The Open Agent Architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1):143–148, 2001.

[Eysenck and Eysenck, 1991] H. J. Eysenck and S. B. G. Eysenck. *The Eysenck Personality Questionnaire-Revised*. Hodder & Stoughton, Sevenoaks, 1991.

[Gill and Oberlander, 2002] Alastair J. Gill and Jon Oberlander. Taking care of the linguistic features of extraversion. In *Proceedings of the 24th Annual Conference of the Cognitive Science Society (CogSci2002)*, pages 363–368, Fairfax, VA, USA, 2002.

[Heylighen and Dewaele, 2002] Francis Heylighen and Jean-Marc Dewaele. Variation in the contextuality of language: An empirical measure. *Foundations of Science*, 7(3):293–340, 2002.

[Minnen *et al.*, 2001] Guido Minnen, John Carroll, and Darren Pearce. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223, 2001.

[Moore *et al.*, 2004] Johanna Moore, Mary Ellen Foster, Oliver Lemon, and Michael White. Generating tailored, comparative descriptions in spoken dialogue. In *Proceedings of the 17th International FLAIRS Conference*, Miami Beach, FL, USA, 2004.

[Oberlander and Gill, 2004] Jon Oberlander and Alastair J. Gill. Individual differences and implicit language: personality, parts-of-speech and pervasiveness. In *Proceedings of the 26th Annual Conference of the Cognitive Science Society (CogSci2004)*, pages 1035–1040, Chicago, IL, USA, 2004.

[Pennebaker *et al.*, 2001] James W. Pennebaker, Martha E. Francis, and Roger J. Booth. *Linguistic Inquiry and Word Count (LIWC): LIWC2001 Manual*. Erlbaum Publishers, Mahwah, NJ, USA, 2001.

[Piwek, 2003] Paul Piwek. A flexible pragmatics-driven language generator for animated agents. In *Proceedings of EACL-03, Research Notes*, Budapest, Hungary, 2003.

[Ratnaparkhi, 1996] Adwait Ratnaparkhi. A Maximum Entropy model for Part-Of-Speech tagging. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, University of Pennsylvania, 1996.

[White and Baldridge, 2003] Michael White and Jason Baldridge. Adapting chart realization to CCG. In *Proceedings of the 9th European Workshop on Natural Language Generation*, pages 119–126, Budapest, Hungary, 2003.

[Wilson, 1988] Michael D. Wilson. The MRC Psycholinguistic Database: Machine readable dictionary, version 2. *Behavior Research Methods, Instruments, & Computers*, 20(1):6–11, 1988.

A Review of Recent Corpus-based Methods for Evaluating Information Ordering in Text Production

Nikiforos Karamanis
Computational Linguistics Research Group
University of Wolverhampton, UK
N.Karamanis@wlv.ac.uk

Chris Mellish
Department of Computing Science
University of Aberdeen, UK
cmellish@csd.abdn.ac.uk

Abstract

This paper surveys the corpus-based methods for evaluating Information Ordering (IO) which emerged recently in the literature on text production. First, we discuss how different assumptions about the input to IO make the preparation of corpora suitable for evaluation more challenging in Natural Language Generation than in Automatic Multidocument Summarisation. Then, we present the types of corpora and performance measures employed by the reviewed work emphasising the considerable consensus that has emerged in these two aspects of automatic evaluation of IO.

1 Introduction

Although evaluating text production systems is much more complicated than evaluating systems analysing text [Hirschman and Mani, 2003], breaking down the tasks that text production consists of may clarify the questions that need to be asked [Mellish and Dale, 1998]. Information Ordering (henceforth, IO), i.e. deciding in which sequence to present a set of preselected information-bearing items is an important issue in Natural Language Generation (NLG) [Reiter and Dale, 2000] and related areas such as Multi-Document Summarisation (MDS) [Barzilay *et al.*, 2002].

Because investigating IO extensively by employing human informants in psycholinguistic experiments is often unfeasible [Karamanis, 2003; Barzilay and Lee, 2004], empirical work on the evaluation of IO has recently become increasingly automatic and corpus-based, which in turn enables large-scale experimentation to take place more easily and researchers to start sharing data between them. Hence, although the corpus-based evaluation of IO is far from mature yet, it seems to represent a good case study on the feasibility of using corpora for testing a specific task in NLG and MDS.

This short paper reviews the corpus-based methods for evaluating IO presented recently in [Duboue and McKeown, 2002], [Dimitromanolaki and Androutsopoulos, 2003], [Lapata, 2003], [Barzilay and Lee, 2004], [Karamanis *et al.*, 2004a; 2004b], [Karamanis and Mellish, 2005] and [Barzilay and Lapata, 2005].[1] Following [Bangalore *et al.*, 2000] who evaluated sentence planning in NLG, all reviewed papers share the assumption that a text production method can be evaluated, albeit approximately, by automatically comparing its output with human-defined solutions attested in a corpus of texts each of which is viewed as a "gold standard". Although we agree (as the other reviewed authors do as well) with [Reiter and Sripada, 2002] that the results of corpus-based evaluation are best treated as hypotheses which eventually need to be integrated with other, perhaps even more time-consuming, evaluation efforts, automatic evaluation might still be particularly useful during development. Thus, reviewing the existing attempts to automatically evaluate IO seems to be justified as it discusses issues worthy of the attention of our colleagues in the text production community.

In the next section, we discuss how the reviewed evaluation efforts differ in the assumptions made about the inputs and outputs of IO depending on whether IO is related to NLG or MDS. We then explain how the different input assumptions introduce additional complications in the preparation of a corpus suitable for NLG-related evaluation. We continue by distinguishing between *multi-authored* and *parallel-authored* corpora employed by the reviewed research and assess which type best meets the *representativeness* requirement [McEnery, 2003]. Last but not least, we present what appears to be the main novelty of the reviewed evaluation efforts, that is, the performance measures they introduce to automatically compare machine-generated orderings of information-bearing items with human-defined orderings from a corpus. We distinguish between *search-oriented* and *distance-based* evaluation measures and discuss which type of measure is more suitable to evaluate certain types of IO methods.[2] The review is completed with a summary of our main conclusions.

[1] [Karamanis *et al.*, 2004a; 2004b; Karamanis and Mellish, 2005] are based on [Karamanis, 2003]. We do not consider work which is primarily based on human judgments such as [Walker *et al.*, 2002].

[2] Although we distinguish between *deterministic* and *non-deterministic* ways (in our terms *methods*) of producing an ordering, we refrain from explaining in detail how the machine-generated orderings which are subject to the reviewed evaluation experiments (in our terms *efforts*) are actually produced.

Inputs to IO:	Surface text	Database facts
Evaluation effort:	MDS-related	NLG-related
Papers:	[Lapata, 2003], [Barzilay and Lee, 2004], [Karamanis *et al.*, 2004b], [Barzilay and Lapata, 2005]	[Duboue and McKeown, 2002], [Dimitrom. and Andr. 2003], [Karamanis *et al.*, 2004a], [Karamanis and Mellish, 2005]

Table 1: Classification of reviewed papers in terms of the assumed input to Information Ordering (IO) and the corresponding type of evaluation effort

2 Information ordering: inputs and outputs

Typically, IO in MDS constitutes a separate module which receives words already organised as sentences [Lapata, 2003; Barzilay and Lee, 2004; Barzilay and Lapata, 2005] (or, less typically, finite clauses [Karamanis *et al.*, 2004b]) as its input. The output of this module is simply an ordering of the input set of information-bearing items. We will subsequently refer to the evaluation efforts in which surface text chunks are assumed to be the input to IO as "MDS-related" (displayed in the second column of Table 1).

The remaining reviewed evaluation efforts (summarised in the third column of Table 1) are characterised as "NLG-related" because they hypothesise an input to IO similar to the representation typically used during an early NLG stage called document planning [Reiter and Dale, 2000]. This input is a system-specific collection of database facts (also known as messages) which (although they are often seen as corresponding to sentences or clauses) are not yet realised as surface text when document planning is performed.

Notably, organising IO as a separate module is not the mainstream NLG view. Rather, in standard NLG, IO is the result of organising the database facts to a tree-like structure during document planning. Crucially, even [Duboue and McKeown, 2002], the only paper in our review that evaluates a hierarchical document planning method, present a corpus-based evaluation measure dealing with orderings rather than tree-like representations. This suggests that traditional document planning methods could also be evaluated in terms of IO, especially since it seems easier to annotate a text for the ordering of the information-bearing items it contains (than the way these items are organised into a hierarchical structure).

However, using database facts as the input to IO introduces additional complications in the preparation of data suitable for NLG-related evaluation, as discussed in the next section in more detail.

2.1 Issues in corpus annotation

The MDS-related evaluation efforts typically test IO methods which rely on syntactic and semantic features that can normally be annotated directly on the input text giving rise to representations similar to the ones employed in other corpus-based research. While [Barzilay and Lapata, 2005] annotate a large collection of texts automatically using methods reported to suffer from an at least 10% error rate, [Karamanis *et al.*, 2004b] make use of a much smaller corpus manually yet very reliably annotated [Poesio *et al.*, 2004]. [Barzilay and Lee, 2004] rely on a knowledge-lean IO method which is applied directly to many un-annotated texts in five different domains.

On the other hand, for a corpus to be used for research

Database fact		**Sentence**
`subclass(ex1, amph)`	→	This exhibit is an amphora.
`painted-by(ex1, p-Kleo)`	→	This exhibit was decorated by the Painter of Kleofrades.
`painter-story(p-Kleo, en4049)`	→	The Painter of Kleofrades used to decorate big vases.
`exhibit-depicts(ex1, en914)`	→	This exhibit depicts a warrior performing splachnoscopy before leaving for the battle.
`current-location(ex1, wag-mus)`	→	This exhibit is currently displayed in the Martin von Wagner Museum.
`museum-country(wag-mus, ger)`	→	The Martin von Wagner Museum is in Germany.

Figure 1: Database facts corresponding to sentences

in NLG, the surface text which represents the system's target output has to be mapped with representations of the input data that the text is supposed to communicate [Reiter and Dale, 2000; Reiter and Sripada, 2002]. This requirement makes the corpora used for the NLG-related evaluation of IO quite different from the corpora typically employed in the MDS-related evaluation efforts.

[Duboue and McKeown, 2002] responded to this requirement by manually mapping each of the texts in their corpus to database facts from their NLG system. Manual annotation is not a trivial effort and, if not properly controlled, the resulting representation might represent several biases introduced by the annotator [Mellish and Dale, 1998].

As the texts of [Duboue and McKeown, 2002] were produced by humans who did not have the system's input in mind, they were found to express information not contained in the system's database while information that the system was supposed to express in a certain text was not verbalised. Moreover, a close examination of their data reveals that e.g. the phrase "John Doe, a 41-year old patient of Dr Smith" (taken from Figure 6 in [Duboue and McKeown, 2002]) maps to at least three facts from the database in a way that makes it difficult to define an order between them.

Problems like these surface quite commonly in NLG research and can be addressed by revising the human text to a more simplified, NLG-tuned (yet usually more stilted) document [Reiter and Dale, 2000]. Instead of that, [Duboue and McKeown, 2002] adopted an evaluation measure which appears to be flexible to account for these discrepancies, at least to a certain extent (see section 4.2). The case remains, however, that extensive misalignments of this kind may severely compromise the NLG-related corpus-based evaluation of IO.

An innovative solution to the aforementioned problems comes from [Dimitromanolaki and Androutsopoulos, 2003] who not only made the human authors aware of the assumed input to IO but also exercised extensive control over the way this input is realised as surface text. First, they derived many sets of facts from the database of their NLG system representing a hypothetical input to IO in their domain. After each fact has been realised as a stand-alone sentence (i.e. without performing any pronominalisation or aggregation), a domain expert was asked to order the sentences in each set and these orderings served as the basis of the evaluation of their IO method. A subset of their data was also employed in the experiments of [Karamanis *et al.*, 2004a] who evaluated different methods for IO, albeit in the same domain. Figure 1

Authoring:	Multi-authored	Parallel-authored
Representativeness:	Assumed	Verified
Papers:	[Duboue and McKeown, 2002], [Barzilay and Lee, 2004], [Karamanis *et al.*, 2004b], [Barzilay and Lapata, 2005]	[Lapata, 2003], [Karamanis and Mellish, 2005]

Table 2: Classification of reviewed papers in terms of authoring methods and representativeness of the employed corpus

(from [Karamanis *et al.*, 2004a]) shows an example set of sentences corresponding to database facts in the order defined by the expert.

Translating facts to sentences makes it feasible to produce a corpus appropriate for NLG-related evaluation by consulting subjects who do not have to become familiar with the internal representation used in a specific system. Moreover, the ordering of sentences defined by the humans can be compared with the output of various IO methods operating on the corresponding facts. However, due to the lack of aggregation and pronominalisation, although this unconventional corpus may consist of well-ordered sentences, it could still be far from fluent and will probably be of limited use for other corpus-based research.

Overall, it seems that using corpora for the MDS-related evaluation of IO is rather straightforward. This is not the case for the NLG-related evaluation efforts due to the different assumptions made about the input to IO. Hence, producing resources which are suitable for the NLG-related corpus-based evaluation requires considerable effort and may give rise to a collection of rather unconventional data.

3 Corpus authoring and representativeness

In addition to the issues related to corpus annotation, another well known problem in the evaluation of text production systems is that the same information may be communicated successfully in many ways [Mellish and Dale, 1998]. As far as IO is concerned it is clear that the ordering attested in the corpus might not be the only good way of organising the underlying information. Although this appears to severely challenge the view of the corpus text as a gold standard, it is also widely acknowledged that presenting information randomly gives rise to a vast number of inappropriate orderings which are very unlikely to manifest in a corpus. Hence, a corpus can provide useful information, especially if it is *representative* [McEnery, 2003], i.e. manifests a range of solutions available to an author. Crucially, evaluation results produced by averaging over a representative corpus are less likely to be biased in favour of a particular IO strategy [Duboue and McKeown, 2002].

Most reviewed studies make use of a *multi-authored* corpus, i.e. texts which are not all written by the same author. The larger the multi-authored corpus, the more representative it is likely to be. Hence, the corpora used for evaluation in [Barzilay and Lee, 2004] and [Barzilay and Lapata, 2005], each of which consists of 100 texts in more than one domain, are expected to provide more coverage than the corpora of [Duboue and McKeown, 2002] and [Karamanis *et al.*, 2004b] (consisting of just 23 and 20 texts respectively).[3] By contrast, although the corpus collected by [Dimitromanolaki and Androutsopoulos, 2003] is large (880 sets of ordered sentences), it is authored by one domain expert only (whom we will henceforth refer to as E0), raising the question whether the ordering preferences on which their evaluation is based on are shared by other experts.

[Karamanis and Mellish, 2005] attempt to verify the generality of the data created by E0. Similarly to [Lapata, 2003], they constructed a *parallel-authored* corpus by asking three more experts to produce additional orderings using the same materials as E0. The measure discussed in section 4.2 was used to investigate the extent to which the experts agree with each other. [Karamanis and Mellish, 2005] report that two out of the three experts they consulted agreed with each other as well as with E0 to a great extent, thus verifying the reliability of the corpus initially collected by [Dimitromanolaki and Androutsopoulos, 2003].

Although disagreement between authors cannot (and in fact should not) always be eliminated [Mellish and Dale, 1998], a well designed study like the one carried out by [Karamanis and Mellish, 2005] is likely to control conditions so that a basis for agreement is found amongst the authors. Like [Lapata, 2003], this study uses human agreement as the upper bound in the corpus-based evaluation of automatic IO methods which again provides an alternative to the view of the corpus text as an absolute gold standard.

Table 2 presents a classification of the reviewed work in terms of the authoring method and representativeness of the employed corpora. Clearly, a parallel-authored corpus represents the range of choices available to an author in a much more straightforward way than a multi-authored corpus, which presumably explains why representativeness has been subject to experimental investigation in the former (but not the latter) case as far as IO is concerned.[4] However, similarly to standard psycholinguistic experiments, consulting many informants can be easily done on a small scale but is more difficult to extend to a larger collection of texts.

Thus, [Karamanis and Mellish, 2005], who had to rely on expert advice only, collected three additional orderings for just 16 sets of sentences from the initial collection of [Dimitromanolaki and Androutsopoulos, 2003] (that is, less than 2% of total). Similarly [Lapata, 2003], who used non-expert informants and the web-based interface of [Barzilay *et al.*, 2002], collected a larger number of parallel sentence orderings (approximately 33 per text) for 12 texts in total which were randomly selected from a corpus consisting of more than 98,000 texts.[5]

[3] However, as already suggested, there is a tradeoff between corpus size and the reliability of the employed annotation methods.

[4] Clearly, the representativeness of a multi-authored corpus can also be investigated e.g. by comparing the choices different authors make in similar situations [Reiter and Sripada, 2002].

[5] [Lapata, 2003] also made use of the parallel-authored corpus collected by [Barzilay *et al.*, 2002] which consists of 10 parallel orderings for a total of 10 texts.

Perf. measure:	search-oriented	distance-based
IO method:	non-deterministic	any method
Papers:	[Karamanis *et al.*, 2004a; 2004b], [Barzilay and Lee, 2004], [Barzilay and Lapata, 2005],	[Duboue and McKeown, 2002], [Lapata, 2003], [Karamanis and Mellish, 2005],

Table 3: Classification of reviewed papers in terms of the employed performance measure and the IO method it may evaluate

4 Devising performance measures

As mentioned in the introduction, perhaps the most novel aspect of the reviewed work is the attempt to evaluate IO quantitatively using a performance measure. To discuss these measures in detail, we first need to distinguish between non-deterministic and deterministic methods for performing IO. Non-deterministic IO methods, assumed by some reviewed evaluation efforts, select the best ordering of information-bearing items among various alternatives on the basis of scores assigned by a metric of coherence [Karamanis *et al.*, 2004a; 2004b] or a probabilistic language model [Barzilay and Lee, 2004; Barzilay and Lapata, 2005]. The aim of the evaluation in this case is to estimate how well the metric (or model) will perform for the purposes of IO even before it is used to produce any actual output. This gives rise to the first type of measures which are search-oriented and estimate *how likely* a non-deterministic IO method is to produce the attested ordering in the corpus as its preferred output.

Most of the remaining evaluation efforts test deterministic IO methods which produce an ordering without explicitly comparing it with its alternatives. Although these cannot be evaluated in terms of the search-oriented measures, their performance can be estimated in terms of distance-based measures which signify *how close* the generated ordering stands when compared with a human defined ordering. Notably, the second set of measures can be applied to the evaluation of non-deterministic methods too as shown in [Karamanis and Mellish, 2005]. Table 3 classifies the reviewed papers in terms of the performance measure employed and the IO method this measure may evaluate.

4.1 Search-oriented evaluation measures

As already mentioned, the corpus-based experiments of [Karamanis *et al.*, 2004a; 2004b] take place prior to the actual generation of an ordering of information-bearing items. Their aim is to identify the most suitable metrics among the many candidates that can be used for the non-deterministic IO method they assume. To distinguish between the metrics, they introduce a performance measure called the *classification rate* of a metric of coherence M on a gold standard ordering (abbreviated here as GSO) which estimates how likely M is to produce the GSO as the output of IO.

To calculate the classification rate (signified as υ), M first assigns a score to the GSO. Then, the space of alternative orderings (defined by the permutations of the sentences that the GSO consists of) is searched. Each alternative ordering is scored and its score is compared with the score given to the GSO. Better(M,GSO) stands for the percentage of orderings that are found to score better than the GSO according to M, whilst Equal(M,GSO) is the percentage of orderings that score equal to the GSO according to M. The formula:

$$\upsilon = Better(M, GSO) + \frac{Equal(M,GSO)}{2}$$

expresses the classification rate of M on the GSO as the expected percentage of alternative orderings which will have a higher chance than the GSO to be selected as the output of IO should M be used to drive this process under the scenario assumed by [Karamanis *et al.*, 2004a; 2004b].[6] The lower the classification rate, the better M is found to perform.

[Karamanis *et al.*, 2004a; 2004b] discuss how the performance of a metric can be estimated using many GSOs from a multi-authored corpus and show how different metrics of coherence can be compared with each other using the classification rate as their performance measure. An experimental methodology is detailed which consists of a set of pairwise comparisons that employ the Sign Test to test significant differences between the metrics.

[Barzilay and Lapata, 2005] and [Barzilay and Lee, 2004] also search the space of alternative orderings of the sentences that the GSO consists of. [Barzilay and Lapata, 2005] introduce a probabilistic ranking model which, like the metrics of [Karamanis *et al.*, 2004a; 2004b], compares the GSO with each of these orderings. The performance measure they introduce is called *ranking accuracy* and expresses the percentage of alternative orderings that are ranked lower than the GSO by their model. In terms of [Karamanis *et al.*, 2004a; 2004b], the ranking accuracy equals 100%−Better(M,GSO).[7]

A different stochastic model is used in [Barzilay and Lee, 2004] to compute the probability of generating the GSO and each alternative ordering in the explored search space. Then, all orderings are ranked according to this probability and the rank given to the GSO is retrieved. The *average GSO rank*, i.e. the average rank given to the GSO across the documents in a certain domain, serves as the first performance measure of their model. The GSO rank reports the mere number of alternative orderings that appear between the GSO and the top ranked ordering for each text without considering how many orderings stand beneath the GSO (which is a variable taken into account in the computation of the classification rate).

For instance, the worst average rank reported in [Barzilay and Lee, 2004] is 15.38 in the "drugs" domain which consists of texts made of 10.3 sentences on average, while the standard deviation of the number of sentences per text in this domain is 7.5 sentences. Assuming that no alternative ordering is assigned the same probability as the GSO, let us first consider the case in which the GSO is ranked as the 15th best permutation in a text that consists of e.g. 5 sentences. The population of alternative orderings in that case is 120 and the resulting classification rate is 12.5%.

However, if the GSO comes from a text consisting of e.g. 10 sentences, the same rank corresponds to a classification rate of less than $2*10^{-7}\%$. Hence, using the classification rate as the performance measure makes it possible to state quite safely that in the second case the relative amount of

[6] A detailed justification of the cited formula in relation to the assumed scenario is presented in chapter 5 of [Karamanis, 2003].

[7] In contrast to [Karamanis *et al.*, 2004a; 2004b], neither [Barzilay and Lapata, 2005] nor [Barzilay and Lee, 2004] consider the possibility of the GSO scoring the same as another ordering.

alternative orderings which score higher than the GSO is much smaller than in the first example (perhaps giving rise to an average performance value across both documents). By contrast, using the GSO rank as the only performance measure does not distinguish between the two cases at all.

While [Barzilay and Lee, 2004] attempt to exhaustively enumerate billions of alternative orderings, the other experimenters deal with much smaller search spaces. As the search space grows factorially, exhaustive enumeration of orderings will unavoidably become impractical. However, [Karamanis *et al.*, 2004a] present detailed arguments that search-oriented evaluation can be practically limited to a sample of 10^6 alternative orderings irrespective of the actual size of the search space.

So far, there has not been any attempt to apply the search-oriented evaluation measures to a parallel-authored corpus. Hence, the generality of some results might be questioned especially when they are based on a small corpus as e.g. in [Karamanis *et al.*, 2004b]. Equally important is the fact that, although [Barzilay and Lee, 2004] and [Barzilay and Lapata, 2005] experiment on reasonably large corpora, they do not report any statistical tests to verify that the observed differences are significant.

To compare their model with the deterministic IO method of [Lapata, 2003], [Barzilay and Lee, 2004] additionally introduce the *GSO prediction rate*, i.e. the percentage of documents in the corpus in which the GSO is ranked first. This measure expresses how often their model reproduces the GSO across the texts in a corpus and does not actually require looking at any other ordering in the search space. However, as the GSO does not always get the highest rank, three rank ranges are also introduced to report the percentage of documents in which the GSO is ranked a) between 1st and 4th b) between 5th and 10th and c) below the 10th position.

In essence, all four search-oriented measures reviewed in this section (classification rate, ranking accuracy, average GSO rank, rank range) are trying to account for the possibility that an IO method might produce an ordering different from the GSO, which can be tested the method is actually used to produce an ordering due to its non-deterministic nature. By penalising the IO method proportionally to its failure to promote the GSO as the best scoring ordering, its performance can be compared with other ways of performing IO within and across domains.

4.2 Distance-based evaluation measures

Deterministic IO methods cannot be evaluated using a search-oriented measure. To evaluate their classification-based IO method [Dimitromanolaki and Androutsopoulos, 2003] measure the percentage of correct selections at each position made by their program compared to the selection made by the human author. However, as they acknowledge, ordering a sequence of sentences is not a series of independent decisions. Hence, calculating precision at each position does not appear to be the most appropriate evaluation measure. By contrast, measures which estimate automatically *how close* the ordering of information-bearing items actually produced by an IO system stands in comparison to the ordering attested in a corpus represent a better solution.

[Duboue and McKeown, 2002] employ a computationally intensive global alignment measure (typically used to compare DNA sequences) which is computed according to the Needleman-Wunsch Algorithm (NWA) as defined in [Durbin *et al.*, 1998]. This measure allows them to compare a computer-generated ordering of facts with the ordering of facts in the corresponding human text even if the latter consists of fewer, more, or even distinct facts. The main problem with the NWA is that it relies on predetermined scores for aligning pairs of facts (or aligning a fact with a gap). As [Duboue and McKeown, 2002] do not discuss how these scores are actually specified, we are not aware of any indication of how this can be done.

[Lapata, 2003] was the first to present an experimental setting which employs the *distance between two orderings* to estimate automatically how close a sentence ordering produced by her probabilistic IO method stands in comparison to an ordering provided by a human judge. Based on the argumentation in [Howell, 2002], [Lapata, 2003] selects Kendall's τ as the most appropriate measure to estimate this distance. Kendall's τ is based on the number of *inversions* between the two orderings and is calculated as follows:

$$\tau = 1 - \frac{2I}{P_N} = 1 - \frac{2I}{N(N-1)/2}$$

P_N stands for the number of pairs of sentences and N is the number of sentences to be ordered.[8] I stands for the number of inversions, that is, the number of adjacent transpositions necessary to bring one ordering to another. Kendall's τ ranges from -1 (inverse ranks) to 1 (identical ranks). The higher the τ value, the smaller the distance between the two orderings.

In the simplest case, [Lapata, 2003] computes the *average* τ score (denoted as T) to compare orderings produced by different versions of her system with the orderings found in a multi-authored corpus and employs the Tukey test to investigate significant differences between the scores.[9]

However, what makes the measure in [Lapata, 2003] particularly appealing is that it has also been applied to parallel-authored corpora which enable one to investigate the range of solutions for IO much more straightforwardly than multi-authored ones. [Lapata, 2003] first computes the average distance between all human defined orderings which represents the upper bound of the evaluation. This score is then compared with the distance between the orderings produced by (different versions of) her model and the human orderings.

Perhaps due to space restrictions, the way that the average T scores are computed is somehow glossed over in [Lapata, 2003]. A more detailed account of this evaluation procedure is presented in [Karamanis and Mellish, 2005] who applied her methodology to verify the generality of the data collected by [Dimitromanolaki and Androutsopoulos, 2003] as already mentioned in section 3. Additionally, [Karamanis and

[8] Like all other reviewed measures (except for the one employed in [Duboue and McKeown, 2002]), Kendall's τ is meant to compare different orderings of identical items.

[9] Table 2 of [Barzilay and Lee, 2004] also reports evaluation results using Lapata's measure, but these are not discussed at all in their "Results" section.

Mellish, 2005] employ T measures for the evaluation of coherence metrics used under a non-deterministic IO method, building upon [Karamanis *et al.*, 2004a].[10]

5 Conclusion

To sum up, the review of the emerging corpus-based evaluations for IO in MDS and NLG leads us to the following conclusions: First, producing a corpus suitable for the NLG-related evaluation of an IO method is considerably demanding because of the need to represent unconventional input data. This is not the case for MDS-related evaluations which rely on features that can be easily annotated.

Despite the differences in input representations, in this review we were able to identify two general types of corpora as well as two types of evaluation measures which appear to be particularly helpful for the automatic evaluation of IO irrespective of whether it takes place within NLG or MDS. We treat this consensus as an indication that the corpus-based evaluation of IO is indeed feasible and might enjoy considerable popularity in future work.

More specifically, it is reasonable to expect that the distinctions and measures discussed in sections 3 and 4 might become standard in the MDS-related corpus-based evaluation of IO. Provided that the problems discussed in section 2.1 are looked at more closely, the same thing could happen in NLG-related evaluation as well.

Acknowledgments

Many thanks to Laura Hasler, Constantin Orasan and Victor Pekar for helpful discussions and to Mirella Lapata for providing us with her latest paper.

References

[Bangalore *et al.*, 2000] Srinivas Bangalore, Owen Rambow, and Steven Whittaker. Evaluation metrics for generation. In *Proceedings of INLG 2000*, pages 1–8, Israel, 2000.

[Barzilay and Lapata, 2005] Regina Barzilay and Mirella Lapata. Modeling local coherence: An entity-based approach. In *Proceedings of ACL 2005*, 2005.

[Barzilay and Lee, 2004] Regina Barzilay and Lillian Lee. Catching the drift: Probabilistic content models with applications to generation and summarization. In *Proceedings of HLT-NAACL 2004*, pages 113–120, 2004.

[Barzilay *et al.*, 2002] Regina Barzilay, Noemie Elhadad, and Kathleen McKeown. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17:35–55, 2002.

[Dimitromanolaki and Androutsopoulos, 2003] Aggeliki Dimitromanolaki and Ion Androutsopoulos. Learning to order facts for discourse planning in natural language generation. In *Proceedings of the 9th European Workshop on Natural Language Generation*, Budapest, Hungary, 2003.

[10] Notably, this is the only domain in which both average T scores on a parallel-authored corpus as well as the classification rate, albeit on single-authored corpus, have been reported.

[Duboue and McKeown, 2002] Pablo Duboue and Kathleen McKeown. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of INLG 2002*, pages 89–96, Harriman, NY, USA, July 2002.

[Durbin *et al.*, 1998] Richard Durbin, Sean Eddy, Anders Krogh, and Greame Mitchinson. *Biological Sequence Analysis*, pages 17-28. Cambridge University Press, 1998.

[Gaizauskas, 1998] Robert Gaizauskas. Evaluation in language and speech technology. *Computer Speech and Language*, 12(4):249–262, 1998.

[Hirschman and Mani, 2003] Linette Hirschman and Inderjeet Mani. Evaluation. In Mitkov [2003], chapter 22, pages 414–429.

[Howell, 2002] David C. Howell. *Statistical Methods for Psychology*. Duxbury, Pacific Grove, CA, 5th edition, 2002.

[Karamanis and Mellish, 2005] Nikiforos Karamanis and Chris Mellish. Using a corpus of sentence orderings defined by many experts to evaluate metrics of coherence for text structuring. In *Proceedings of ENLG05*, Aberdeen, UK, 2005. Poster session.

[Karamanis *et al.*, 2004a] Nikiforos Karamanis, Chris Mellish, Jon Oberlander, and Massimo Poesio. A corpus-based methodology for evaluating metrics of coherence for text structuring. In *Proceedings of INLG04*, pages 90–99, Brockenhurst, UK, 2004.

[Karamanis *et al.*, 2004b] Nikiforos Karamanis, Massimo Poesio, Chris Mellish, and Jon Oberlander. Evaluating centering-based metrics of coherence using a reliably annotated corpus. In *Proceedings of ACL04*, pages 391–398, Barcelona, Spain, 2004.

[Karamanis, 2003] Nikiforos Karamanis. *Entity Coherence for Descriptive Text Structuring*. PhD thesis, Division of Informatics, University of Edinburgh, 2003.

[Lapata, 2003] Mirella Lapata. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of ACL 2003*, pages 545–552, Saporo, Japan, July 2003.

[McEnery, 2003] Tony McEnery. Corpus linguistics. In Mitkov [2003], chapter 24, pages 448–463.

[Mellish and Dale, 1998] Chris Mellish and Robert Dale. Evaluation in the context of natural language generation. *Computer Speech and Language*, 12(4):349–373, 1998.

[Mitkov, 2003] Ruslan Mitkov, editor. *The Oxford Handbook of Computational Linguistics*. Oxford University Press, 2003.

[Poesio *et al.*, 2004] Massimo Poesio, Rosemary Stevenson, Barbara Di Eugenio, and Janet Hitzeman. Centering: a parametric theory and its instantiations. *Computational Linguistics*, 30(3):309–363, 2004.

[Reiter and Dale, 2000] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

[Reiter and Sripada, 2002] Ehud Reiter and Somayajulu Sripada. Should corpora texts be gold standards for NLG? In *Proceedings of INLG 2002*, pages 97–104, Harriman, NY, USA, July 2002.

[Walker *et al.*, 2002] Marilyn A. Walker, Owen Rambow, and Monica Rogati. Training a sentence planner for spoken dialogue using boosting. *Computer Speech and Language*, 16:409–433, 2002.

Using an Annotated Corpus As a Knowledge Source For Language Generation

Tomasz Marciniak and Michael Strube
EML Research gGmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
http://www.eml-research.de/nlp

Abstract

We present an approach to building a realization component for an NLG system that makes use of an annotated corpus as the source of linguistic knowledge. Three issues that we focus on include: modeling of the generation decisions, corpus annotation and specification of the system architecture.

1 Introduction

Linguistic realization, the *tactical* part of Natural Language Generation (NLG), can be defined as a mapping between the conceptual content of a linguistic expression and its grammatical form. To perform this mapping, an NLG system requires a substantial amount of knowledge of how lexical and syntactic constructions can be applied at different levels of the linguistic organization (i.e. *clause*, *discourse*) to encode the intended meaning.

In this paper we present an approach to building a trainable realization component applicable to narrow domains such as *route directions*. Instead of relying on hand-crafted linguistic resources, such as a grammar or a lexicon, the system acquires all the lexical and syntactic knowledge necessary for linguistic realization from a small, domain-specific corpus. Hence, the task of collecting a corpus and annotating it with the required semantic and grammatical information becomes an integral part of system development.

The methodology we propose assumes three stages: modeling of the generation decisions, preparation of the corpus and specification of the processing flow. The first step involves a semantic and grammatical analysis of the target texts, and its goal is to identify low-level decisions that would drive the generation process. A corpus of relevant target texts must be then collected and annotated with the required semantic and grammatical information. The final step is to determine the architecture that would specify the flow of information between modules handling individual decisions. We compare two such architectures: a sequential model which assumes a strict ordering of generation decisions and an integrated model, based on an Integer Linear Programming formulation.

The paper is structured as follows: in Section 2 we discuss the use of corpora in NLG. Sections 3 and 4 are concerned respectively with modeling the semantics and grammatical form of route directions. In Section 5 we describe the annotation process and in Section 6 we introduce two implemented architectures and report on the evaluation results.

2 The Use of Corpora in NLG

The main function of a corpus in generation is to provide a basis for *requirements analysis* [Reiter and Dale, 2000]. It aims at determining the target texts, characterizing their structure and identifying domain-specific linguistic constructions used to communicate the input data. In the traditional *rule-based* approach to generation, the information gathered in this way can be used by system developers to explicitly model the behavior of the system.

Alternatively, a corpus can be treated as a source of data from which the generation knowledge can be extracted using statistical and machine learning techniques. This approach can be used to construct trainable components which do not require extensive qualitative analysis and domain expertise on the part of the developer and help to overcome the knowledge acquisition bottleneck. Using this approach, the developer cannot influence directly the choices that the system makes. It is the quality of the corpus and the right formulation of the generation decisions that determine the quality of the output texts (cf. [Reiter *et al.*, 2003]). The advantage of using corpus-based techniques, is that they can be ported to new domains and applications. They also help to bridge the gap between aspects of generation traditionally viewed as separate, such as sentence vs. discourse generation, or lexicalization vs. syntactic realization.

2.1 Related Work

Corpus-based methods were introduced to NLG in the context of syntactic realization [Langkilde and Knight, 1998]. Most current works in this area follow the *ranking* approach which involves rule-based overgeneration and then corpus-driven selection of the best candidate, e.g. [Bangalore and Rambow, 2000b; Varges and Mellish, 2001]. Empirical techniques were also applied to solving isolated tasks, such as lexical choice, e.g. [Bangalore and Rambow, 2000a], ordering of NP modifiers, e.g. [Shaw and Hatzivassiloglou, 1999] or sentence ordering, e.g. [Lapata, 2003]. In these works, corpus-based methods were applied to solving individual tasks in a non-application context. To the best of our knowledge, a few

projects only tried to cover the whole process of linguistic realizations relying exclusively on trainable components [Chen *et al.*, 2002; Kan and McKeown, 2002].

2.2 Our Goals

In the current work we investigate the use of a single corpus as the sole source of linguistic knowledge necessary to drive the process of linguistic realization. In formal terms, we can define this process as a one-to-many mapping between a representation of the semantic content of an expression and its linguistic form. We concentrate on three important issues that development of a corpus-based realization component involves:

- abstract modularization of the generation process in terms of discrete, low-level tasks whose realization is to be learned from the corpus,
- annotation of the corpus with the semantic and grammatical information, corresponding respectively to the input and output of the system, and
- specification of the system architecture which determines the processing flow.

We illustrate this three-step development process on the example of *route directions* generation.

3 Route Directions

To identify the target texts in our domain we first performed an informal analysis of human-authored route directions available on the Internet. We identified three categories of such texts, characterized by different levels of the linguistic and conceptual complexity:

1. Directions consisting of phrasal expressions only, with no verbs or discourse connectives, e.g. *Down the street, past the post office, left onto Church Street*.
2. Directions expressed with imperative clauses only, with a limited number of verbs and discourse connectives, e.g. *Leave the building, turn right onto Dowman, go to 24th Street.*
3. Directions expressed with complex texts, characterized by a broad lexical (verbs, connectives) and structural variation, (see Example 1 below).

   Example 1. (a) Standing in front of the hotel (b) follow Meridian street south for about 100 meters, (c) passing the First Union Bank entrance on your right, (d) until you see the river side in front of you. (e) Then make a left onto North Hills Street. (f) The auditorium will be up the street on your left.

We decided to collect a corpus of texts from category (3) for two reasons[1]. Firstly, we wanted to ensure that the generator can express complex content in a consistent manner and hence decided not to mix different types of texts (cf. [Reiter and Sripada, 2002]). Secondly, we noticed structural similarities between route directions from (3) and other types of instructional texts, such as cooking or assembly instructions, which offered a promise of reusing, at least partially, the annotated corpus.

In our application, the task of generating route directions comprises two stages. The *strategic* part is concerned with mapping a topological specification of a route, i.e. a *route plan* onto a dynamic model organized around a set of temporally related *situations*. In the domain of route directions, situations schematize the spatio-temporal interactions between the route follower and salient elements of the environment (e.g. streets, landmarks). The *tactical* part of the generation process, on which we focus in this paper, consists in constructing the linguistic description of the underlying content.

3.1 Ontology of Situations

The ontology of situations constitutes a semi-formal specification of the conceptual content of route directions, and as such sanctions the input to the realization component. It spans three conceptual levels, each having its own taxonomy of classes and properties, related to one another through axioms specifying well-formedness conditions (see [Marciniak and Strube, 2005b] for a detailed discussion).

At the *aspectual level*, a situation is assigned to a specific aspectual category such as: *state*, *process*, *accomplishment* or *achievement* [Vendler, 1967]. The membership in a given category is further formalized as a function of three binary attributes: *stative, durative* and *culminated* [Moens and Steedman, 1988].

At the *frame level*, the conceptual category of each situation is specified. In our domain, three main categories are *SelfMotion*, *Localization* and *VisualPerception*. Each of them is modeled as a frame, with slots specifying conceptual roles attributed to situation participants (see Table 1)[2].

Frame	*Conceptual Roles*
SELFMOTION	self_mover, source, path, goal, etc.
VISUALPERCEPTION	perc_object, location, direction, etc.
LOCALIZATION	loc_object, location, direction, etc.

Table 1: Frames with corresponding conceptual roles

Finally, situations are not isolated, but occur in temporally structured groups. Three temporal relations that we consider are: *initialRelation*, *ongoingRelation* and *subsequentRelation*. The relations are functional and non-reversive, which results in the structure taking the form of a tree (see Figure 1). For each pair of related situations, the *child node* situation is recognized as the *trajector* being located relative to the *landmark* situation (i.e. during its *initial*, *ongoing* or *subsequent* stage). This trajector-landmark asymmetry finds direct manifestation in the corresponding linguistic form. In Example 1, for instance, situation denoted by (d) bears the *linguistic marking* (i.e. discourse connective), which signals its relation to the situation from (b)[3]. Hence, to properly realize a linguistic description of a situation, its temporal context must be considered.

[1] Notice, however, that there is no clear-cut boundary between categories (2) and (3).

[2] The categories are partially modeled after FrameNet *frames* and roles correspond to *frame elements*, cf. [Baker *et al.*, 1998])

[3] Temporal relations can be also signaled by the verb form e.g. *gerund* in (a,c), or simply the linear ordering, e.g. (d) $\prec$ (e)

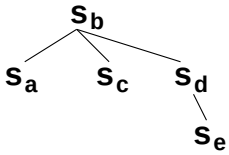

Figure 1: Temporal structure of situations. Nodes s_a, s_b, etc. correspond to discourse units (a), (b), etc. from Example 1.

4 Modeling Generation Decisions

Linguistic realization of route directions comprises a range of different types of generation decisions. At the clause level, verbs and prepositions must be lexicalized, and the syntactic frame of the clause must be specified[4]. At the discourse level, clauses must be ordered, and where appropriate, discourse connectives lexicalized. To be able to learn these decisions from a corpus, we first need to determine their scope[5], and then formulate them as classification problems.

In the approach presented here, we associate low-level generation decisions with *minimal* elements of the grammatical form, which may affect the meaning of an expression or render it ill-formed.

4.1 LTAG-based Representation

To determine the range of generation decisions relevant to our application, we first model the grammatical form of route directions using Lexicalized Tree Adjoining Grammar (LTAG) [Joshi and Schabes, 1991]. LTAG provides a useful abstraction of the process of building the grammatical form of an expression, called *derivation*. It starts with the selection of *elementary trees*, anchored by lexical items, such as verbs or prepositions. In the next step, elementary trees are assembled by means of well-defined operations of *substitution* and *adjunction*. Originally used to model the sentence derivation only, LTAG has been shown to apply equally well to the structure of a discourse (cf. [Webber and Joshi, 1998]). At the discourse level, each clause in a text is associated with an elementary tree anchored by a discourse connective. Following the underlying temporal or rhetorical relations, discourse-level trees are combined to form the derived structure.

As an example, consider the derivation of discourse unit (c) from Example 1 in the context of (a), (b) and (d) (Figure 2). At the clause level, the set of elementary trees includes one *initial* tree α_1 anchored by the main verb which projects to *S* and specifies the syntactic frame of the clause, and *auxiliary* trees β_1 and β_2 corresponding to the verb arguments. The auxiliary trees are successively adjoined to *VP* node of α_1 immediately dominated by *S*. The order in which the adjunctions take place determines the final ordering of the arguments in the clause.

At the discourse level (cf. Figure 3), the discourse unit describing the root situation from Figure 1 is modeled as the initial tree α_2, and auxiliary trees β_3 and β_4 represent the related discourse units. Following the temporal dependencies,

[4]In our application, nominal expressions are own names mostly, e.g. *Meridian Street*, and as such are directly specified in the input, hence we ignore here the problem of NP generation.

[5]E.g. decide whether specifications of the lexical and grammatical form of a verb constitute a single decision, i.e. *passing* or if they are determined separately: *pass + gerund*

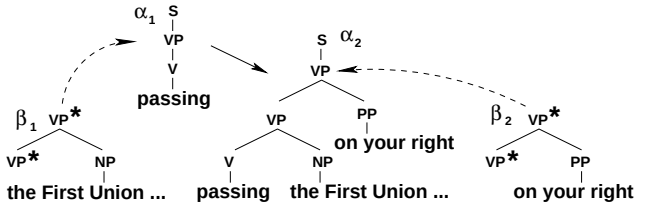

Figure 2: Clause-level derivation

trees β_3 and β_4 are adjoined to α_2. Again, the ordering of the operations determines the linear structure of the text.

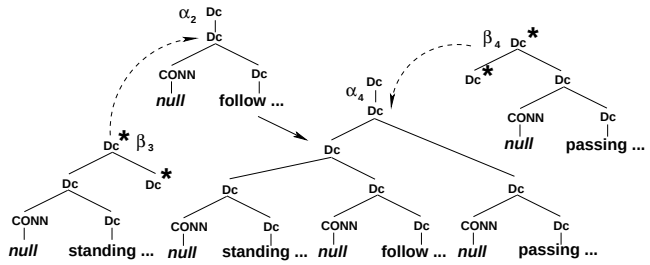

Figure 3: Discourse-level derivation

4.2 Feature Vector Encoding

To model the whole process in a uniform way we represent elementary trees with feature-value pairs necessary to account for the distinctions between individual trees and determine the order of adjunction operations (see Table 2).

Elementary Tree	Feature	Possible Values
Main Verb	*s_exp*	*NP, NP-VP*
	verb_lex	*walk, follow, pass, etc.*
	verb_form	*bare_inf, gerund, etc.*
Verb Argument	*adj_rank*	*numeric*
	phr_type	*NP, PP, P*
	prep_lex	*to, past, towards, etc.*
Discourse Unit	*adj_rank*	*numeric*
	adj_dir	*left, right*
	connective	*null, and, until, etc.*

Table 2: Frames with corresponding conceptual roles

The tree associated with the main verb in a clause is represented with three features: *s_exp, verb_lex* and *verb_form*, denoting respectively whether a clause should have an explicit subject (i.e. *NP-VP*), the lexical form of the verb and its grammatical form. For each verb argument, the corresponding tree is modeled with features: *adj_rank* which determines the ordering of the adjunction operations, *phr_type* which specifies the syntactic category of the argument, and *prep_lex*, i.e. the lexical form of the preposition/particle (if *phr_type* is other then *NP*). Finally, the discourse-level trees representing discourse units are modeled with three features: *adj_rank* and *adj_dir* denoting respectively the ordering of the adjunction operations and the adjunction direction and *connective* specifying the lexical form of the discourse connective (or *null* if no explicit connective is present).

4.3 Generation Decisions Revisited

Realizations of individual features used to encode the LTAG derivation are now taken to constitute single generation deci-

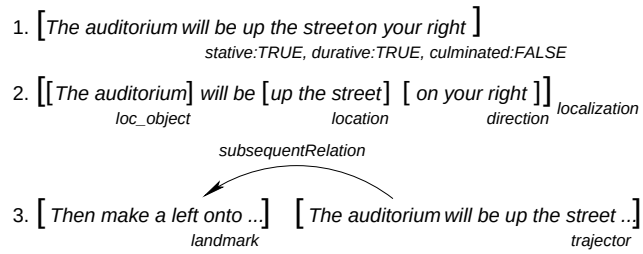

Figure 4: Three levels of annotation: 1. *Aspectual Level*, 2. *Frame Level* and 3. *Temporal Level*

sions. Each such decision constitutes a classification problem and consists in assigning a class label to an instance specifying the semantic context of the particular decision.

Notice that this formulation draws no distinctions between lexical and syntactic decisions on the one hand, and discourse-level and clause-level decisions on the other. This fact simplifies the procedure of acquiring the necessary knowledge from the corpus. For each decision, a classifier handling a single-valued prediction needs to be *induced* from a set of training instances.

5 Corpus Annotation

To obtain the training data, we collected a corpus of 75 route directions texts, with a total number of 904 discourse units. By applying the concept of *stand-off* annotation, i.e. separation of tags from the actual texts, we were able to realize the annotations at multiple levels. Each annotation level comprises a set of *markables*, i.e. marked text spans associated with the required linguistic information and falling in four categories:

- *Discourse Unit* markables, corresponding to individual clauses,
- *Predicate* markables, corresponding to main verbs within clauses,
- *Argument* markables, corresponding to phrasal arguments of verbs, and
- *Connective* markables, corresponding to conjunctions and discourse markers.

During the pre-processing stage the texts were tokenized, POS-tagged, and markables were automatically detected with a simple, rule-based system, tuned to the given type of texts.

5.1 Semantic Annotation

In order to apply the ontological model of situations to the annotation task we defined an annotation scheme comprising a selection of semantic tags which provide a flat representation of the categories specified in the ontology (Table 3).

Annotations have been realized at different levels, corresponding to the levels specified in the ontology (Figure 4). At the *aspectual* level, each *Discourse Unit* markable has been tagged with three boolean attributes: *stative, durative* and *culminated*. At the *frame* level, *Discourse-unit* markables have been tagged with frame labels (i.e. *self_motion, localization, etc.*) and *Argument* markables have been assigned semantic roles (e.g. *source, path* or *goal*). Finally, at the *temporal* level, pairs of *Discourse Unit* markables corresponding to related situations have been respectively tagged as *trajector* and *landmark* and linked by a directed relation carrying a specific label (i.e. *initialRelation, ongoingRelation* or *subsequentRelation*). In Table 4 we provide frequencies with which respective attributes were assigned to the corresponding markables.

Frame Str.	*Freq.*	*Aspect. Str.*	Freq.	*Temp. Str.*	*Freq.*
self_motion	739	*stative*	129	*initialRel.*	92
localization	114	*durative*	432	*ongoingRel.*	235
vis_perc.	51	*culminated*	539	*culminatedRel.*	481

Table 4: Frequencies of semantic attributes at different annotation levels.

Conn.	*Freq.*	*S-Exp.*	*Freq.*	*Verb Form*	*Freq.*	*Verb Lex.*	*Freq.*
null	494	*NP-VP*	213	*bare_inf*	480	*walk*	107
and	158	*VP*	691	*fin_pres*	146	*turn*	91
until	56			*gerund*	106	*continue*	59
as	37			*will_inf*	67	*pass*	57
then	22			*to_inf*	9	*null*	53

Table 5: Frequencies of selected LTAG-based attributes.

5.2 Grammatical Annotation

The goal of grammatical annotation was to impose the LTAG-based encoding on the texts from the corpus. Since discourse units, main verbs, arguments and connectives were labeled during the pre-processing stage, the corresponding lexical and ordering attributes became readily available, and hence were left implicit. The only explicitly tagged attribute was *verb_form*, associated with *Predicate* markables. Frequencies of selected attributes are given in Table 5.

6 Processing Model

The abstract architecture of our system comprises a set of n classification tasks $T = \{T_1, ..., T_n\}$. Each task T_i consists in assigning a label from $L_i = \{l_{i1}, ..., l_{im_i}\}$ to an instance representing the particular generation decision. To solve individual tasks, machine learning classifiers are trained on a set of data extracted from the annotated corpus.

Since in order to generate a well-formed and fluent text, individual generation decisions cannot be handled in isolation, the final issue to be determined during implementation, concerns the problem of integrating individual generation decisions with one another. This is a long-standing problem in NLG (cf. e.g. [Reiter, 1994]), and amounts to determining the flow of information between tasks. One advantage of casting different types of generation decisions in a single classification-oriented format, is that it is easy to implement and test them in various configurations. In the rest of this section we describe two alternative models, a sequential and an integrated one, and give the results of the initial, quantitative evaluation.

6.1 Sequential Model

In the pipeline system, implemented as a *cascade of classifiers*, the output representation is built incrementally, with subsequent classifiers having access to the outputs of the previous modules. Figure 5 illustrates this type of processing

Ontological Level	*Semantic Tags*	*Markable Level*
Aspectual Level.	*stative, durative, culminated*	*Discourse-unit*
Frame Level	*self_motion, visual_perception, localization*	*Discourse-unit*
	self_mover, source, path, goal, direction, distance ...	*Argument*
Temporal Level	*trajector, landmark*	*Discourse-unit*
	initialRelation, ongoingRelation, subsequentRelation	*Discourse-unit*

Table 3: Semantic annotation scheme

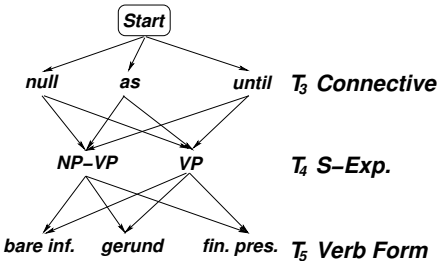

Figure 5: Sequential model on the example of three tasks: *Connective, S-Exp.* and *Verb Form*

as a traversal of a multi-layered lattice, with the individual layers corresponding to single tasks, and the nodes representing the respective outcomes. At each step, a corpus trained classifier outputs a probability distribution, and the transition augmented with the highest probability is selected. A well known problem with this type of processing is that generation decisions are dependant on one another and hence the initial decisions lack the necessary contextual information provided by the those occuring later (cf. [Danlos, 1984]).

6.2 Integrated Model

This problem is apparently eliminated in an integrated architecture, with all decisions being handled within a single process. As shown in Figure 6, an integrated process can be modeled as a graph, with the nodes representing outputs from individual tasks, and the edges marking inter-dependencies. The problem amounts to finding a path through the graph, so that for each task, a single node is selected, and *global* distribution of costs associated with both selected nodes and transitions is considered.

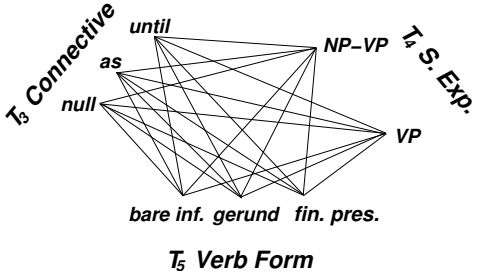

Figure 6: Integrated model

6.3 Linear Programming Formulation

To solve this graph search problem we apply an optimization technique, called Integer Linear Programming (ILP). An ILP problem consists of decision variables $x_1, x_2, ..., x_n$ augmented with *costs* and combined in a linear target function, and a set of constraints blocking *illegal decisions*. The goal is to find such an assignment of the variables that the decision function is either maximized or minimized within the bounds provided by the constraints (see e.g. [Nemhauser and Wolsey, 1999]). In our application, we use *binary* variables to model both nodes in the graph, corresponding to single labels, and transitions between nodes. The objective function is then expressed as a weighted sum of the variables[6]:

$$min \;\; c(l_{11})x(l_{11}) + c(l_{12})x(l_{12}) + ... + c(l_{ij}l_{kp})x(l_{ij}, l_{kp})$$

The constraints we add specify that firstly, variables may take binary values only, i.e. $x \in 0, 1$, secondly, for each task only one variable may be selected, and finally, if two variables $x(l_{ij})$ and $x(l_{kp})$ modeling a pair of labels belonging to two different tasks are selected, then also $x(l_{ij}, l_{kp})$ co-modeling this pair of labels must be selected[7].

6.4 Evaluation

We evaluated both systems using *leave-one-out* cross-validation, i.e. each text was used once for testing, and the remaining texts provided the training data. We used *Naive-Bayes* classifier to learn realizations of individual tasks from the data, and to solve the ILP model we applied *lp_solve*, a GNU-licence Mixed Integer Programming (DIP) solver[8].

The goal of the evaluation was to see how good both systems are at mirroring the human-authored texts. To assess such defined performance we applied three metrics: accuracy and *Kappa* to evaluate individual tasks and *Phi*, a distance measure that we used to compare the feature-based representations of the generated texts and the original ones. The results given in Table 6 show that both systems reached relatively high scores, with *Kappa* over 70% for almost all tasks and the end-to-end score *Phi* lying over 85%. This proves that the *meaning-to-form* mapping can be successfully learned from a relatively small corpus. In addition, an improvement of the ILP system over the pipeline in almost all tasks and the overall score *Phi* indicates that an integrated architecture offers a serious advantage over the sequential model.

7 Conclusions

In this paper we presented a corpus-based approach to building a trainable realization component for an NLG system. We concentrated on three important aspects of this task: abstract representation of the generation process in terms

[6] Costs of single nodes $c(l_{ij})$ are calculated as $-log(P(l_{ij}))$, where $P(l_{ij})$ is the probability of selecting label l_{ij} for task T_i, output by the classifier. Costs of transitions $c(l_{ij}l_{kp})$ are given by $-log(P(l_{ij}, l_{kp}))$, with $P(l_{ij}, l_{kp})$ denoting the joint probability of labels l_{ij} and l_{kp} co-occuring in the corpus.

[7] For details on the ILP formulation see [Marciniak and Strube, 2005a]

[8] http://www.geocities.com/lpsolve/

Tasks	Pipeline Accuracy	Pipeline κ	ILP Accuracy	ILP κ
Adj Rank	96.81%	90.90%	97.43%	92.66%
Adj. Dir.	98.04%	89.64%	97.95%	89.05%
Connective	79.10%	61.14%	79.36%	61.31%
S Exp.	96.20%	90.17%	99.49%	98.65%
Verb Form	87.83%	78.90%	93.22%	88.30%
Verb Lex	67.40%	64.19%	76.08%	74.00%
Phrase Type	87.08%	73.36%	88.03%	77.17%
Prep. Lex	86.95%	81.12%	88.59%	83.24%
Adj Rank	86.95%	78.65%	91.27%	85.72%
Phi	0.87		0.90	

Table 6: Results of quantitative evaluation of the ILP and pipeline systems.

of classification-oriented decisions, annotation of the corpus with the necessary linguistic information and aggregation of individual decisions within a single integrated architecture. In the evaluation we showed that a corpus-based realization in a narrow domain presents itself as a feasible task requiring a small amount of annotated data only.

Acknowledgements: The work presented here has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author receives a scholarship from KTF (09.001.2004).

References

[Baker *et al.*, 1998] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, Montréal, Québec, Canada, 10–14 August 1998, pages 86–90, 1998.

[Bangalore and Rambow, 2000a] Srinivas Bangalore and Owen Rambow. Corpus-based lexical choice in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, China 7–12 July 2000, 2000.

[Bangalore and Rambow, 2000b] Srinivas Bangalore and Owen Rambow. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics*, Saarbrücken, Germany, 31 July – 4 August 2000, pages 42–48, 2000.

[Chen *et al.*, 2002] John Chen, Srinivas Bangalore, Owen Rambow, and Marilyn Walker. Towards automatic generation of natural language generation systems. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 24 August – 1 September, 2002, 2002.

[Danlos, 1984] Laurence Danlos. Conceptual and linguistic decisions in generation. In *Proceedings of the 10th International Conference on Computational Linguistics*, Stanford, Cal., pages 501–504, 1984.

[Joshi and Schabes, 1991] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars and lexicalized grammars. In *Maurice Nivat and Andreas Podelski, editors*, Definability and Recognizability of Sets of Trees. *Elsevier*, 1991.

[Kan and McKeown, 2002] M.Y. Kan and K. R. McKeown. Corpus-trained text generation for summarization. In *Proceedings of the 2nd International Conference on Natural Language Generation*, New York, NY, 1-3 July, 2002, 2002.

[Langkilde and Knight, 1998] Irene Langkilde and Kevin Knight. 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*, Montréal, Québec, Canada, 10–14 August 1998, pages 704–710, 1998.

[Lapata, 2003] Mirella Lapata. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, 7–12 July 2003, pages 545–552, 2003.

[Marciniak and Strube, 2005a] Tomasz Marciniak and Michael Strube. Beyond the pipeline: Discrete optimization in NLP. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, Ann Arbor, MI, 29-30 June, 2005, pages 136–143, 2005.

[Marciniak and Strube, 2005b] Tomasz Marciniak and Michael Strube. Modeling and annotating the semantics of route directions. In *Proceedings of the 6th International Workshop on Computational Semantics*, Tilburg, The Netherlands, January 12-14, 2005, pages 151–162, 2005.

[Moens and Steedman, 1988] Marc Moens and Mark Steedman. Temporal ontology and temporal reference. *Computational Linguistics*, 14:15–28, 1988.

[Nemhauser and Wolsey, 1999] George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization*. Wiley, New York, NY, 1999.

[Reiter and Dale, 2000] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, UK, 2000.

[Reiter and Sripada, 2002] Ehud Reiter and Somayajulu Sripada. Should corpora texts be gold standards for NLG? In *Proceedings of the 2nd International Conference on Natural Language Generation*, New York, NY, 1-3 July, 2002, pages 97–104, 2002.

[Reiter *et al.*, 2003] Ehud Reiter, Somayajulu Sripada, and Roma Robertson. Acquiring correct knowledge for natural language generation. *Journal of Artificial Intelligence Research*, 18:491–516, 2003.

[Reiter, 1994] Ehud Reiter. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the 7th International Workshop on Natural Language Generation*, Kennebunkport, MA, 21-24 June 1994, pages 160–173, 1994.

[Shaw and Hatzivassiloglou, 1999] James Shaw and Vasileios Hatzivassiloglou. Ordering among premodifiers. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, Maryland, July 1999, pages 135–143, 1999.

[Varges and Mellish, 2001] Sebastian Varges and Chris Mellish. Instance-based natural language generation. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, PA, 2-7 June, 2001, pages 1–8, 2001.

[Vendler, 1967] Zeno Vendler. Verbs and times. In *Linguistics in Philosophy*, Cornell University Press, Ithaca, NY, pages 97–121, 1967.

[Webber and Joshi, 1998] Bonnie Lynn Webber and Aravind Joshi. Anchoring a lexicalized tree-adjoining grammar for discourse. In *Proceedings of the COLING/ACL '98 Workshop on Discourse Relations and Discourse Markers*, Montréal, Québec, Canada, 15 August 1998, pages 86–92, 1998.

The Penn Discourse TreeBank as a Resource for Natural Language Generation

Rashmi Prasad, Aravind Joshi, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki
Institute for Research in Cognitive Science, University of Pennsylvania
rjprasad,joshi,nikhild,aleewk,elenimi@linc.cis.upenn.edu
Bonnie Webber
Division of Informatics, University of Edinburgh
bonnie@inf.ed.ac.uk

Abstract

While many advances have been made in Natural Language Generation (NLG), the scope of the field has been somewhat restricted because of the lack of annotated corpora from which properties of texts can be automatically acquired and applied towards the development of generation systems. In this paper, we describe how the Penn Discourse TreeBank (PDTB) can serve as a valuable large scale annotated corpus resource for furthering research in NLG and for inducing models for the development of NLG systems. The PDTB is annotated for discourse relations, and encodes explicitly the elements of these relations: explicit and implicit discourse connectives, denoting the predicates of the relations, and text spans, denoting the arguments of the relations. Connectives and arguments are also annotated with features and spans related to attribution, and each connective will be annotated with labels standing for the projected discourse relation, including sense distinctions for polysemous connectives. We exemplify the use of the corpus for two tasks in NLG: the realization of discourse relations during sentence planning, and the representation and realization of attribution.

1 Introduction

Many NLG systems, especially those that follow the pipelined architecture [Reiter, 1994] comprising modules for *content determination* (aka *text planning*), *microplanning* (aka *sentence planning*) and *surface realization*, do not specify discourse relations between elementary content units, ECUs (i.e., propositions denoting the simplest predication over entities), in the output of the text planner (e.g., [Rambow and Korelsky, 1992; Walker *et al.*, 2001]). Some that do represent discourse relations, do so by using (pre-defined) schemas - broadly following [McKeown, 1985] - and assume a one-to-one mapping between discourse relations and discourse connectives (e.g., [Davey, 1979; Hovy, 1987]).[1] The shortcoming of such systems is that their flexibility is severely restricted by the schemas because of which they are also not easily portable to other domains.

In contrast, an increasing number of systems have extended their text planning components to represent discourse relations in the text plan – e.g., [Hovy, 1993; Mellish *et al.*, 1998; Walker *et al.*, 2003]. However, with the lack of a complete understanding of discourse relations and of the ways they can be realized in text, what such systems now require are corpus resources from which to derive knowledge needed by the next module in the pipeline (*sentence planning*) to model the interaction of aggregation and discourse relations.[2] In this paper, we discuss what the PDTB [Miltsakaki *et al.*, 2004; Prasad *et al.*, 2004; Webber *et al.*, 2005] can contribute to natural language generation, focusing on the sentence planning task of discourse relation lexicalization (DR-lexicalization), including *occurrence*, *selection*, and *placement* [Moser and Moore, 1995], and on the representation of attribution in the text plan as well as its interaction with aggregation.

In Section 2, we give a brief overview of the Penn Discourse TreeBank annotations. In Section 3 we discuss the relevance of the PDTB for sentence planning tasks, addressing the DR-lexicalization problems of *occurrence*, *selection*, and *placement* in detail. In Section 4, we discuss the attribution annotations in the PDTB and show how they can be useful for the representation of attribution for content determination, as well as for their realization during sentence planning. We summarize in Section 5.

2 The Penn Discourse TreeBank

The PDTB contains annotations of *explicit* and *implicit* discourse connectives and their arguments on the 1 million word Wall Street Journal corpus. Following the views toward discourse structure in [Webber *et al.*, 2003], the PDTB treats discourse connectives as discourse-level predicates that take **two** *abstract objects* such as events, states, and propositions

[1] [Rambow and Korelsky, 1992] also use schemas but do not represent discourse relations between the units.

[2] Here, as for the rest of this paper, we define aggregation in its broadest possible sense, to stand for any syntactic transformation that can be applied to two *lexicalized* content units (CUs), including the PERIOD operation, which is applied to generate two input CUs as two sentences, as well as the (non-)realization of discourse relations. For the purpose of this paper, we assume that content units are lexicalized before aggregation, but that discourse relations are lexicalized during or after aggregation.

[Asher, 1993] as their arguments. For example, in (1), the subordinating conjunction *since* is a discourse connective that establishes a TEMPORAL relation between the event of the earthquake hitting and a state where no music is played by a certain woman.[3]

(1) *She hasn't played any music* since **the earthquake hit**.

The following four classes of explicit connectives are annotated in the PDTB (Examples provided for each class are only a few of those annotated in the PDTB - see Section 2.4.):

- *subordinating conjunctions*, both bare (e.g., *because*, *when*, *since*, *although*) and with a modifier (e.g., *only because*, *particularly since*, *even after*),
- *subordinators* (e.g., *in order that*, *except that*),[4]
- *coordinating conjunctions* (e.g., *and*, *or*, *nor*), and
- *discourse adverbials* (e.g., *however*, *otherwise*, *then*).[5,6]

Because there are, as yet, no generally accepted abstract semantic categories for classifying the arguments to discourse connectives as have been suggested for verbs (e.g., *agent*, *patient*, *theme*, etc.), the two arguments to a discourse connective are simply labelled *Arg2*, for the argument that appears in the clause that is syntactically bound to the connective, and *Arg1*, for the other argument. In examples used in this paper, the text whose interpretation is the basis for *Arg1* appears in italics, while that of *Arg2* appears in bold. For the subordinating conjunctions, since the subordinate clause is bound to the connective, *Arg2* corresponds to the subordinate clause, and hence the linear order of the arguments can be *Arg1-Arg2* (Ex. 2), *Arg2-Arg1* (Ex. 3), or *Arg2* may appear embedded in *Arg1* (Ex. 4), depending on the relative position of the subordinate clause with respect to its governing matrix clause.

(2) *Third-quarter sales in Europe were exceptionally strong*, boosted by promotional programs and new products – although **weaker foreign currencies reduced the company's earnings**.

(3) Michelle lives in a hotel room, and although **she drives a canary-colored Porsche**, *she hasn't time to clean or repair it.*

(4) *Most oil companies*, when **they set exploration and production budgets for this year**, *forecast revenue of $15 for each barrel of crude produced*.[7]

The order of the arguments for adverbials and coordinating conjunctions is typically *Arg1-Arg2* since *Arg1* usually appears in the prior discourse. But as Example (5) shows, the arguments of discourse adverbials *can* appear embedded within one another. In this example, *Arg1* is embedded in *Arg2*.

(5) As an indicator of the tight grain supply situation in the U.S., market analysts said that **late Tuesday the Chinese government**, *which often buys U.S. grains in quantity*, **turned** instead **to Britain to buy 500,000 metric tons of wheat**.

Abstract objects can be arbitrarily complex in the PDTB so that arguments of connectives can be associated with single clauses, multiple clauses, single sentences, or multiple sentences. However, a *minimality principle* requires an argument to contain the minimal amount of information needed to complete the interpretation of the relation. Any other span of text that is perceived to be relevant (but not necessary) in some way to the interpretation of arguments is annotated as *supplementary information*, labelled *Sup1*, for material supplementary to *Arg1*, and *Sup2*, for material supplementary to *Arg2*.

Also as a consequence of the abstract object characterization of arguments, arguments may be denoted by non-clausal units such as *nominalizations* that have an event interpretation, and *discourse deictics* (*this*, *that*) that refer to abstract objects.

2.1 Implicit Connectives

Implicit connectives are annotated in the PDTB between adjacent sentences when no connective appears explicitly to relate the second sentence to the first. For example, in (6), the second sentence is related to the first via an EXPLANATION relation (i.e, Mr. Breeden's wise perception of the ways of Washington is being used as an explanation for the assertion that he may be able to succeed), but this relation is not expressed explicitly.

(6) *Also unlike Mr. Ruder, Mr. Breeden appears to be in a position to get somewhere with his agenda*. IMPLICIT=BECAUSE **As a former White House aide who worked closely with Congress, he is savvy in the ways of Washington**.

Annotation at such points consists of a record of an explicit connective that "best" conveys the implicit relation perceived as holding between the adjacent sentences. For the implicit relation perceived in Example (6), *because* is recorded as the connective. In order to account for multiple "simultaneous" relations between the same two abstract objects, there may also be more than one connective between the sentences. Example (7) shows an annotation where two relations were perceived as holding simultaneously, and for which the connectives *when* and *for example* were recorded.

[3] The assumption of the arity constraint of the arguments has been upheld in all the annotation done thus far. Discourse-level predicate-argument structures are therefore unlike the predicate-argument structures of verbs at the sentence-level (PROPBANK [Kingsbury and Palmer, 2002]), where verbs can take any number of arguments.

[4] The class of subordinators was added at a later stage.

[5] *Discourse adverbials* are to be distinguished from *clausal adverbials* (see [Forbes, 2003]).

[6] Discourse markers such as *well*, *anyway*, *now*, etc., that signal the organizational or focus structure of the discourse, are not annotated.

[7] As this example shows, annotations in the PDTB can be discontinuous, a feature allowed by *WordFreak*, the discourse annotation tool (developed by Tom Morton and Jeremy Lacivita). Discontinuous annotation is possible for connectives as well, such as for *on the one hand . . . on the other hand.*

(7) *The small, wiry Mr. Morishita comes across as an outspoken man of the world.* IMPLICIT=WHEN IMPLICIT=FOR EXAMPLE ($_{sup2}$ Stretching his arms in his silky white shirt and squeaking his black shoes) **he lectures a visitor about the way to sell American real estate and boasts about his friendship with Margaret Thatcher's son**.

As examples (6) and (7) show, the annotation of implicit connectives also includes the marking of the textual span from the two adjacent sentences that are the arguments of the inferred implicit relation. That the spans selected for the two arguments need not (trivially) constitute the entire sentence can be seen in Example (7).

At the current stage of the project, implicit connectives between adjacent sentences across paragraphs, and intra-sentential connectives (such as those occurring with *free adjuncts*) are not annotated.

2.2 Sense Annotation

All explicit and implicit connectives in the PDTB will be annotated with labels for the discourse relation that they denote, including sense distinctions for polysemous connectives (e.g., *since, while, if, when, because*). For example, *since* seems to have three different senses, one purely TEMPORAL (as in Ex. 8), another purely CAUSAL (as in Ex. 9) and a third both CAUSAL and TEMPORAL (as in Ex. 10).

(8) *The Mountain View, Calif., company has been receiving 1,000 calls a day about the product* since **it was demonstrated at a computer publishing conference several weeks ago**.

(9) *It was a far safer deal for lenders* since **NWA had a healthier cash flow and more collateral on hand**.

(10) *. . . and domestic car sales have plunged 19%* since **the Big Three ended many of their programs Sept. 30**.

2.3 Attribution Annotation

Attribution, which has to do with ascribing beliefs and assertions expressed in text to the agent(s) holding or making them, is annotated in the PDTB to primarily distinguish between two different sources of attribution, the Writer of the text ("Writer attribution"), or some other Speaker (or Agent) mentioned by the Writer ("Speaker Attribution"). With respect to attribution associated with discourse connectives and their arguments, there are broadly two possibilities:[8]

Case 1 A discourse connective and both its arguments are attributed to the same source, either the Writer, as in Example (1), or the Speaker (Bill Biedermann) in Example (11):

(11) "*The public is buying the market* when **in reality there is plenty of grain to be shipped**," said Bill Biedermann, Allendale Inc. research director.

Case 2 One or both arguments have a different attribution value from the discourse connective. In Example (12), the connective and *Arg1* are attributed to the Writer, whereas *Arg2* is attributed to another Speaker (here, the purchasing agents):[9]

(12) *Factory orders and construction outlays were largely flat in December* while purchasing agents said **manufacturing shrank further in October**.

Attribution tags in the PDTB are currently being further refined (while still maintaining the basic distinction between Speaker and Writer attribution) to include further distinctions between, for example, verbs of saying and verbs of propositional attitude, and to represent the interaction of verbs of attribution with negation and factuality. In addition, the second release of the PDTB will also record the text span associated with the source and type of attribution. For further discussion of attribution annotation in PDTB, see [Dinesh *et al.*, 2005].

2.4 Summary and Project Goals

The first release of the PDTB (November 2005) will contain approx. 16K annotations of explicit connectives (approx. 6000 subordinating conjunctions, 5000 discourse adverbials and 5000 coordinating conjunctions) and approx. 20K annotations of implicit connectives. There are over 90 different types of connectives.

3 PDTB and Sentence Planning

Following the introduction of *sentence planning* as an independent intermediate stage [Rambow and Korelsky, 1992] in the traditional two-way split of NLG systems into a content determination component and a realization component, discourse connectives have invited a great deal of research in NLG, as they are related simultaneously to the *aggregation* and *lexicalization* tasks in sentence planning. Assuming the basic three-way pipelined architecture [Reiter, 1994], the input to the sentence planning component is, thus, taken to be a hierarchically ordered text plan structure that encompasses all the elementary content units (ECUs) that the system has decided to generate, as the leaves of the structure, with the internal nodes specifying the discourse relations holding between ECUs or groups of ECUs.[10]

In the pipelined architecture, sentence planning is relieved of decisions related to content determination, specifically that of determining the discourse relation that holds between the CUs, so that it can focus on the problem of how to *express* the discourse relations. Work on connective usage [Moser and Moore, 1995] has identified three separate but related decision making processes during sentence planning, for the generation of discourse connectives: (a) *occurrence*, i.e., whether to generate a connective or not; (b) *selection*, i.e., which connective to generate; and (c) *placement*, i.e., where to place the connective.

[8] Attribution is annotated for both explicit and implicit connectives.

[9] When attribution of a connective or its arguments is uncertain, the attribution is assigned to the Writer as a Default.

[10] The structure of the text plan is the same irrespective of whether the text planning task is done in a schema-based top-down manner [McKeown, 1985] or in a bottom-up manner [Marcu, 1997].

While many insightful studies have been carried out on discourse connectives for generation purposes, they have either singled out a few connectives (e.g., [Elhadad and McKeown, 1990; Dorr and Gaasterland, 1995; Rösner and Stede, 1992], or proposed heuristics based on a small number of constructed examples (e.g., [Scott and Souza, 1990]), or proposed classification-based lexicons that are very hand-intensive to build, (e.g., [Grote and Stede, 1998; Knott and Mellish, 1996]), especially in a multilingual context. In contrast, corpora annotated with information about connectives can provide a useful knowledge source from which to automatically induce properties of connectives designing sentence planning tasks. In the rest of this section, we discuss how the PDTB annotations of discourse connectives and their arguments can be useful towards the three tasks of connective generation discussed above.

3.1 Occurrence

As corpus studies [Moser and Moore, 1995; Williams and Reiter, 2003] have shown, more often than not, there is **no** discourse connective explicitly connecting a clause to the previous discourse. When combining two CUs during aggregation, the sentence planner has to make a choice about whether to generate a connective or not. The two important questions to ask here is whether there are significant constraints on the lexicalization of discourse relations, and whether some or all of these constraints can be identified directly from annotated corpora. For example, what is the reason for lexicalizing the CONSEQUENCE relation in (13) and not in (14)?

(13) *The three men worked together on the so-called Brady Commission, headed by Mr. Brady, which was established after the 1987 crash to examine the market's collapse*. As a result **they have extensive knowledge in financial markets, and financial market crises**.

(14) From 1984 to 1987, its (Iverson's) earnings soared six-fold, to $3.8 million, on a seven-fold increase in revenue, to $44.1 million. *But in 1988, it ran into a buzz saw: a Defense Department spending freeze*. IMPLICIT=AS A RESULT **Iverson's earnings plunged 70% to $1.2 million**.

Some research has shown that the choice of whether or not to lexicalize a relation is indeed governed by constraints that can be built into a sentence planner: while some constraints may require deep reasoning over world knowledge and properties of the content units, some are more directly associated with the surface properties of the content units and the text plan. For example, on the one hand, [Amsili and Rossari, 1998] show that in French, the use of a connective to express a CAUSAL relation between eventualities can be constrained by the interaction of the (Vendlerian) *aspectual classes* of the two eventualities as well as the order in which the eventualities appear in the CAUSAL relation. On the other hand, corpus-based research [Williams and Reiter, 2003] has shown that there are statistically significant differences across *classes of connectives* with respect to *how frequently they are lexicalized*.

Like other corpus based work, the PDTB also offers the opportunity to find statistically significant patterns. However, it also offers much more, since, unlike previous studies, the size of the corpus is much larger, and since other layers of annotation on the same text are also available, namely the syntactic annotation of the Penn TreeBank and the semantic annotation of the PropBank. A generation system with an architecture such as the one we have assumed here, provides a syntactic and semantic analysis of the content units to the sentence planning component, so models induced from the PDTB will provide a much richer set of constraints to constitute the criteria for the (non-)lexical occurrence of discourse relations.

Finally, for the occurrence task, it is also useful that the PDTB specially identifies cases where there is no explicit phrase that can be inserted in place of the purported implicit relation between adjacent sentences. These cases were analyzed and distinguished as three types: (a) NOREL, where no discourse relation was inferred between the adjacent sentences (Ex. 15), (b) NOCONN-ENT, where the relation was perceived to be one established by elaboration via entity description (Ex. 16), and finally, (c) NOCONN, where some relation - other than the entity elaboration relation - was perceived, but for one of several reasons, including redundancy, the use of an explicit connective in the text sounded unacceptable (Ex. 17).[11]

(15) *The transaction has been approved by Kyle's board, but requires the approval of the company's shareholders*. IMPLICIT=NOREL **Kyle manufactures electronic components**.

(16) *C.B. Rogers Jr. was named chief executive officer of this business information concern*. IMPLICIT=NOCONN-ENT **Mr. Rogers, 60 years old, succeeds J.V. White, 64, who will remain chairman and chairman of the executive committee**.

(17) *In the 1920s, a young schoolteacher, John T. Scopes, volunteered to be a guinea pig in a test case sponsored by the American Civil Liberties Union to challenge a ban on the teaching of evolution imposed by the Tennessee Legislature*. IMPLICIT-NOCONN **The result was a world-famous trial exposing profound cultural conflicts in American life between the "smart set," . . . and the religious fundamentalists,** . . .

In sum, we hope that the implicit connective annotations in the PDTB will encourage research and experiments that will support decisions related to connective occurrence in NLG.

3.2 Selection

The problem of lexical choice for connectives is well recognized: a given discourse relation can be expressed with a variety of connectives, but there are subtle syntactic, semantic, pragmatic, and stylistic factors that preclude the use of any of a class of connectives in a given context.

[11] One reason that an explicit connective might sound redundant is if the relation is already lexicalised elsewhere in the clause - for example, in the subject (as in Ex. 17) or the verb, as in "This [*resulted in, led to*] a world-famous trial . . . ."

The design of the PDTB annotations provides direct access to the discourse connectives and their arguments (since the annotations are anchored on the connectives), and together with the other layers of annotation (PTB and PropBank), can allow inferences to be drawn easily from the observed patterns. Claims made in the literature about particular connectives can also be empirically tested. In some studies that we have conducted, there are indications that some well-known accounts of connectives are not supported by the PDTB annotations. For example, [Elhadad and McKeown, 1990] argue that while the CAUSE relation can be expressed by both *because* and *since*, the two connectives are not freely interchangeable, and that they differ in whether the information is known to the receiver or not, in that *because* introduces *new* information whereas *since* presents *given* information. Crucially, their account is based on an elsewhere claimed tendency [Quirk *et al.*, 1972] for *because* and *since* to be in complementary distribution, with *because* appearing postposed and *since* appearing preposed, and the notion that *new information* tends to be placed towards the end of a clause [Halliday, 1985]. The PDTB annotations show that while *because* does tend to appear postposed, (see [Prasad *et al.*, 2004]), the 90 confirmed instances of CAUSAL *since* are distributed equally in pre- and postposed position, suggesting a clarification of the above correlation between information status and clause order.

In addition, in earlier work [Prasad *et al.*, 2004], we had integrated a subset of the PDTB annotations with the PTB syntactic annotations, and found that *although* and *even though*, which denote a CONCESSION relation and are thought to be undifferentiable except for *even though* carrying "emphasis" [Huddleston and Pullum, 2002], behaved quite differently with respect to the relative position of their arguments. *Although* clauses were more frequently preposed, whereas *even though* clauses were more frequently postposed. To this, we have now added results obtained for *though*. Table 1 shows the argument-order distribution for *although*, *even though*, and *though*.[12]

CONN	*Arg2* Postposed	*Arg2* Preposed	Total
although	129 (37%)	218 (63%)	347
even though	77 (75%)	26 (25%)	103
though	97 (70%)	42 (30%)	139
Total	303 (51%)	286 (49%)	589

Table 1: Argument Order for *although*, *even though*, and *though*.

Table 1 shows that *even though* and *though* pattern alike, and that their variation with *although* is highly significant. The former occur postposed about 72% of the time and preposed about 28% of the time, the opposite of *although* (see Table 2.) Further analysis of these connectives is needed to determine what the variation might correlate with.[13]

Some recent studies (e.g., [Hutchinson, 2005]) have tried to model the substitutability of discourse connectives based on corpus data. However, the model uses only lexical cooccurrences. We believe that better models could be obtained with corpora such as the PDTB that are aligned with other levels of (syntactic and semantic) analysis. This would be especially beneficial for generation approaches (such as is assumed here) that carry out the task of lexical choice for connectives *after* the abstract syntactic specification for the connective's arguments have already been constructed. This means that syntactic and semantic features of the CUs can play a role in modeling the use of connectives.

[12]The tokens for *though* exlcude its adverbial occurrences.

[13]The similar behavior of *even though* and *though* also suggests that the claim about "emphasis" being the sole distinguishing feature might still stand, but only for these two connectives.

CONN	*Arg2* Postposed	*Arg2* Preposed	Total
although	129 (37%)	218 (63%)	347
(even) though	174 (72%)	68 (28%)	242
Total	303 (51%)	286 (49%)	589

Table 2: Argument Order for *although* and *(even) though*, with *though* and *even though* combined.

3.3 Placement

When relating two CUs, discourse connectives are syntactically bound to one of the CUs (called *Arg2* in the PDTB), so the sentence planner needs to make a decision about (a) which CU to associate the connective with, and (b) where to place the connective in the CU.

The first decision can mostly be made very simply by reference to the relative order of the CUs and the syntactic class of the connective, if it is assumed that linear ordering of the CUs during aggregation is done prior to DR-lexicalization (see Footnote 2). For instance, in both Examples (18) and (19),[14] a CONCESSION relation holds between the two CUs, in that the assertion of John being smart denies the expectation raised by the other assertion, that John is not smart. However, in each case, the linear ordering of the CUs is taken as given for the placement task, and the decision of where to place the connective depends on the relative position of the CU that raises the expectation to be denied, and the syntactic class of the connective selected. If *although*, a subordinating conjunction, is selected, it must be associated with the CU that raises the expectation, whereas if *but*, a coordinating conjunction, is selected, it must be associated with the CU that denies the expectation.

(18) Although John failed the exam, he is smart.

(19) John failed the exam but he is smart.

The second decision, however, is more difficult to make, and relates to discourse adverbials. Unlike subordinating conjunctions and coordinating conjunctions, which can modify their (*Arg2*) CU clause only in initial position, discourse adverbials can occur in several positions in the clause. The examples below show the connective *as a result* appearing in initial position (Ex. 20), in medial position (Ex. 21), and in final position (Ex. 22) in the clause.

[14]These examples are adapted from [Elhadad and McKeown, 1990].

(20) *Despite the economic slowdown, there are few clear signs that growth is coming to a halt*. As a result, **Fed officials may be divided over whether to ease credit**.

(21) *The chief culprits*, he says, *are big companies and business groups that buy huge amounts of land "not for their corporate use, but for resale at huge profit.*" . . . **The Ministry of Finance**, as a result, **has proposed a series of measures that would restrict business investment in real estate** . . .

(22) *Polyvinyl chloride capacity "has overtaken demand* and **we are experiencing reduced profit margins** as a result.", . . .

In previous work [Prasad *et al.*, 2004], we conducted experiments on 5 adverbials (*as a result*, *instead*, *nevertheless*, *otherwise*, and *therefore*), looking at the position in which the connective was realized in the *Arg2* CU clause, and we found that the connectives in this set occurred predominantly in initial position in their clause. However, most of the examples collected for these experiments had the connective in initial position, so we want to re-run the experiments when we have further data.

4 PDTB and the Representation and Realization of Attribution

Taking the theoretical view of language as goal-driven communication, most working NLG systems are built within restricted domains with clearly defined communicative goals. This means that while some tasks, such as some aspects of sentence planning and realization, are modeled in a general way and can be extended across applications, other tasks such as content determination and text planning are driven by the needs of the domain, in particular the information content of the domain. One of the first tasks of NLG systems is thus *domain modeling*, i.e., analyzing target texts and declaring the different types of information that need to be conveyed within the domain, and at what level of granularity. For example, in the restaurant review domain [Walker *et al.*, 2003], the primary kind of entity is *restaurant* with properties like *food*, *service*, and *atmosphere* defined over these entities. In contrast, the weather domain [Reiter and Dale, 2000] includes *time-span* entities, with properties like *rainfall* defined over these entities.

In the News domain, applications that deal with the generation of News reports have to model more complex types of entities and relations, such as the relation of attribution, which is a relation of "ownersip" between abstract objects and individuals or agents. Because they are News reports, writers of these texts are concerned with ascribing beliefs held and statements made to the correct sources (including themselves), and relatedly, with preventing the false inference (on the part of the reader) that a certain piece of information that is being conveyed is a commonly known fact or commonly held view). For example, there is a big difference between the way the basic information in the following two sentences is presented, because it leads the reader of the report to make different inferences about the facts.

(23) The chief culprits, he (Mr. Lee) says, are big companies and business groups that buy huge amounts of land "not for their corporate use, but for resale at huge profit."

(24) The chief culprits are big companies and business groups that buy huge amounts of land "not for their corporate use, but for resale at huge profit."

In Example (23), the assertion that the chief culprits are big companies and business groups is understood as "fact", but from Mr. Lee's point of view.[15] On the other hand, Example (24) strongly suggests an interpretation where the same assertion is considered to be a well-known fact (and hence to be "true").

Even though the attribution relation in the domain model is expressed between all abstract objects and agent entities that are related in this way, the relation information need not necessarily continue on to successive stages of content determination and text planning, at least not in the same manner. For one thing, the content planner may decide to present some information as well-known fact even though it was attributed to some agent. This is somewhat evident from the texts themselves, where many sentences suggest that the writer holds a strong belief that the information from some external source is true, or is confident about its truth, and has decided to drop the attribution, i.e, not realize it. This is a common idea in the definition of ECUs in generation systems, where going from the information contained in the domain model to the definition of the ECUs involves making decisions about what to convey and what not to convey (depending on the overall and immediate communicative goals), especially since the idea is that all pieces of the defined ECUs have to be realized in one way or another.

Secondly, even if attributions are included in the ECUs for realization, the content planner can, and must, have a way of representing them in at least two different ways. Evidence for this comes from the PDTB annotation of attribution on discourse connectives and their arguments. Assuming that the text plan encodes the ECUs and the discourse relations holding between them, a discourse relation may hold either between the attributions themselves or just between the abstract object arguments of the attribution. These two possiblities are shown in Examples (25) and (26):

(25) When **Mr. Green won a $240,000 verdict in a land condemnation case against the state in June 1983**, he says *Judge O'Kicki unexpectedly awarded him an additional $100,000*.

(26) *Advocates said the 90-cent-an-hour rise, to $4.25 an hour by April 1991, is too small for the working poor*, while **opponents argued that the increase will still hurt small business and cost many thousands of jobs**.

[15] In the examples in this section, the text span associated with attribution is shown boxed for illustrative purposes only: the guidelines for annotating attribution spans have not been decided yet.

In Example (25), the TEMPORAL discourse relation is expressed between the eventuality of Mr. Green winning the verdict and the Judge giving him an additional award. The discourse relation does not entail the interpretation of the attribution relation. On the other hand, Example (26) shows that the CONTRAST relation holds between the agent arguments of the attribution relation, which means that the attribution relation must be part of the contrast as well. These examples therefore show that attribution has both an "owner" and a scope, and that both must be correctly represented in the text plan, for appropriate realization.

The attribution relation can be defined over discourse relations as well, as seen in Example (27), where the TEMPORAL relation between the two arguments is presumably also being quoted and thus attributed to some Speaker (some managing director). If discourse relations can indeed be objects of attribution, then it suggests a further extension of the abstract object ontology (in the domain model) and its representation by the content planner.

(27) "When **the airline information came through**, *it cracked every model we had for the marketplace*," said a managing director at one of the largest program-trading firms.

With the above examples, we have illustrated one aspect of the annotation in the PDTB that has a bearing on how NLG systems working within the News report domain must model and represent attribution. The completed attribution annotations in the PDTB corpus will provide a useful resource as a target corpus to confirm the above hypotheses, and to discover other aspects relevant to the representation of attribution.

Carrying on from the stages of domain modeling and content determination, the PDTB annotation shows that the attribution relation continues to affect sentence planning decisions as well. The attribution relation can be realized in a variety of ways: Speaker attribution of a connective and both its arguments can involve either quoted or indirect speech, as in Examples (28) and (29), respectively.

(28) "Now, Philip Morris Kraft General Foods' parent company is committed to the coffee business and to increased advertising for Maxwell House," says Dick Mayer, president of the General Foods USA division. "Even though **brand loyalty is rather strong for coffee**, *we need advertising to maintain and strengthen it*."

(29) Like other large Valley companies, Intel also noted that *it has factories in several parts of the nation*, so that **a breakdown at one location shouldn't leave customers in a total pinch**.

Example (28) also shows that the attribution may be unrealized when the abstract object argument of the attribution is expressed in direct quotes, leaving it to the reader to recover the attribution anaphorically from the preceding sentence.

Finally, Example (30) shows that both arguments of the CONTRAST relation signaled by *nevertheless* are attributed to a Speaker, Mr. Robinson, whereas the relation itself is attributed to the Writer.

(30) Mr. Robinson . . .said *Plant Genetic's success in creating genetically engineered male steriles doesn't automatically mean it would be simple to create hybrids in all crops*. . . . Nevertheless, he said, **he is negotiating with Plant Genetic to acquire the technology to try breeding hybrid cotton**.

Space does not permit us to present the many more ways in which the attribution relation is realized in the PDTB. We hope that the examples provided here will encourage researchers to exploit the PDTB to automatically discover the full range of variation that seems to be present, and model the conditions under which the different realizations are generated. Apart from the interaction of attribution with aggregation, since the PDTB annotation will contain the span of text associated with attribution, it will also provide a valuable resource for determining the different ways in which different types of attribution can be lexicalized, for example, for the choice between different verbs of saying, such as *say* and *note*, and to model the conditions under which one may be used as against the other.

5 Summary

In this paper, we have described how the Penn Discourse TreeBank (PDTB), a large-scale multi-layered annotated corpus of discourse relations, can contribute as a resource towards research and development in NLG. We gave an overview of the types of annotation in the PDTB: explicit and implicit discourse connectives and their arguments, sense distinctions for connectives, and attribution associated with connectives and their arguments. We showed the relevance and use of the annotations for the sentence planning tasks of occurrence, selection, and placement, giving results from experiments conducted on the PDTB, and showing how the results could be integrated into an NLG sentence planner. We also showed that empirical research on the PDTB can be used to test theoretical claims made in the literature about discourse connectives. We then described the role of attribution for domain modeling of applications that deal with generation of WSJ style texts and demonstrated the utility of the PDTB annotations for the representation and realization of attribution.

Acknowledgements

The Penn Discourse TreeBank project is partially supported by NSF Grant: Research Resources, EIA 02-24417 to the University of Pennsylvania (PI: A. Joshi).

References

[Amsili and Rossari, 1998] Pascal Amsili and Corinne Rossari. Tense and connective constraints on the expression of causality. In *Proc. COLING-ACL*, pages 48–54, 1998.

[Asher, 1993] Nicholas Asher. *Reference to Abstract Objects*. Kluwer, Dordrecht, 1993.

[Davey, 1979] Anthony Davey. *Discourse Production*. Edinburgh Univ. Press, 1979.

[Dinesh *et al.*, 2005] Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. Attribution and the (non)-alignment of syntactic and discourse arguments of connectives. In *Proc. ACL Workshop on Frontiers in Corpus Annotation II*, 2005.

[Dorr and Gaasterland, 1995] Bonnie J. Dorr and Terry Gaasterland. Selecting tense, aspect and connecting words in language generation. In *Proc. IJCAI*, pages 1299–1305, 1995.

[Elhadad and McKeown, 1990] Michael Elhadad and Kathleen R. McKeown. Generating connectives. In *Proc. COLING*, volume 3, pages 97–101, 1990.

[Forbes, 2003] Katherine Forbes. *Discourse Semantics of S-modifying Adverbials*. PhD thesis, Univ. of Penn., 2003.

[Grote and Stede, 1998] Brigitte Grote and Manfred Stede. Discourse marker choice in sentence planning. In *Proc. INLG*, pages 128–137, 1998.

[Halliday, 1985] Michael A.K. Halliday. *An Introduction to Functional Grammar*. Edward Arnold, London, 1985.

[Hovy, 1987] Eduard Hovy. *Generating Natural Language under Pragmatic Constraints*. PhD thesis, Yale Univ., 1987.

[Hovy, 1993] Eduard H. Hovy. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63:341–385, 1993.

[Huddleston and Pullum, 2002] Ronald Huddleston and Geoffrey Pullum. *The Cambridge Grammar of the English Language*. Cambridge Univ. Press, Cambridge, UK, 2002.

[Hutchinson, 2005] Ben Hutchinson. Modeling the substitutability of discourse connectives. In *Proc. ACL*, 2005.

[Kingsbury and Palmer, 2002] Paul Kingsbury and Martha Palmer. From TreeBank to PropBank. In *Proc. LREC*, 2002.

[Knott and Mellish, 1996] Alistair Knott and Chris Mellish. A feature-based account of the relations signalled by sentence and clause connectives. *Language and Speech*, 39(2-3):143–183, 1996.

[Marcu, 1997] Daniel Marcu. From local to global coherence: a bottom-up approach to text planning. In *Proc. AAAI*, pages 629–635, 1997.

[McKeown, 1985] Kathleen R. McKeown. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge Univ. Press, Cambridge, U.K., 1985.

[Mellish *et al.*, 1998] Chris Mellish, Mick O'Donnell, Jon Oberlander, and Alistair Knott. An architecture for opportunistic text generation. In *Proc. INLG*, pages 28–37, 1998.

[Miltsakaki *et al.*, 2004] Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. Annotating discourse connectives and their arguments. In *Proc. HLT/NAACL Workshop on Frontiers in Corpus Annotation*, pages 9–16, 2004.

[Moser and Moore, 1995] Megan G. Moser and Johanna D. Moore. Using discourse analysis and automatic text generation to the study of cue usage. In *Proc. AAAI Symposium on Empirical Methods in Discourse Interpretation and Organization*, pages 92–98, 1995.

[Prasad *et al.*, 2004] Rashmi Prasad, Eleni Miltsakaki, Aravind Joshi, and Bonnie Webber. Annotation and data mining of the Penn Discourse Treebank. In *Proc. ACL Workshop on Discourse Annotation*, pages 88–95, 2004.

[Quirk *et al.*, 1972] Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. *A Grammar of Contemporary English*. Longman, London, 1972.

[Rambow and Korelsky, 1992] Owen Rambow and Tanya Korelsky. Applied text generation. In *Proc. ANLP*, pages 40–47, 1992.

[Reiter and Dale, 2000] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge Univ. Press, 2000.

[Reiter, 1994] Ehud Reiter. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proc. INLG*, pages 163–170, 1994.

[Rösner and Stede, 1992] Dietmar Rösner and Manfred Stede. Customizing RST for the automatic production of technical manuals. In R. Dale, E. Hovy, D. Rösner, and O. Stock, editors, *Aspects of Automated Natural Language Generation. Proc. INLG*, pages 199–214, Heidelberg, 1992. Springer.

[Scott and Souza, 1990] Donia R. Scott and Clarisse Sieckenius de Souza. Getting the message across in RST-based text generation. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 47–73. Academic Press, 1990.

[Walker *et al.*, 2001] Marilyn Walker, Owen Rambow, and Monica Rogati. SPoT: A trainable sentence planner. In *Proc. NAACL*, pages 17–24, 2001.

[Walker *et al.*, 2003] Marilyn Walker, Rashmi Prasad, and Amanda Stent. A trainable generator for recommendations in multimodal dialogue. In *Proc. EUROSPEECH*, pages 1697–1701, 2003.

[Webber *et al.*, 2003] Bonnie Webber, Aravind Joshi, Matthew Stone, and Alistair Knott. Anaphora and discourse structure. *Computational Linguistics*, 29(4):545–587, 2003.

[Webber *et al.*, 2005] Bonnie Webber, Aravind Joshi, Eleni Miltsakaki, Rashmi Prasad, Nikhil Dinesh, Alan Lee, and Kate Forbes. A short introduction to the Penn Discourse TreeBank. In *Copenhagen Working Papers in Language and Speech Processing*. 2005.

[Williams and Reiter, 2003] Sandra Williams and Ehud Reiter. A corpus analysis of discourse relations for natural language generation. In *Proc. Corpus Linguistics*, pages 899–908, 2003.

Statistically Generated Summary Sentences: A Preliminary Evaluation using a Dependency Relation Precision Metric

Stephen Wan[12] **Robert Dale**[1] **Mark Dras**[1]
[1]Centre for Language Technology
Div. of Information Communication Sciences
Macquarie University
Sydney, Australia
swan,rdale,madras@ics.mq.edu.au

Cécile Paris[2]
[2]Information and Communication Technologies
CSIRO
Sydney, Australia
Cecile.Paris@csiro.au

Abstract

Often in summarisation, we are required to generate a summary sentence that incorporates the important elements of a related set of sentences. In this paper, we do this by using a statistical approach that combines models of *n*-grams and dependency structure. The approach is one in which words are recycled and re-combined to form a new sentence, one that is grammatical and that reflects the content of the source material. We use an extension to the Viterbi algorithm that generates a sequence that is not only the best *n*-gram word sequence, but also best replicates component dependency structures taken from the source text. In this paper, we describe the extension and outline a preliminary evaluation that measures dependency structure recall and precision in the generated string. We find that our approach achieves higher precision when compared to a bigram generator.

1 Introduction

The state-of-the-art in summarisation technology still rests heavily on the use of sentence extraction techniques. However, recent work has shown that sentence extraction is not sufficient to account for the scope of written human abstracts ([Jing and McKeown, 1999]; [Knight and Marcu, 2002]). In our own work on United Nations Humanitarian Aid Proposals, we noticed that only 30% of sentences from human authored abstracts could be extracted from the source document. Often non-extracted summary sentences are the result of recombining words and phrases from selected source document sentences. The end product is either a paraphrase or a fusion of information from component sentences.

Thus, to generate abstract-like summaries, we are motivated to explore a procedure that allows for a recombination of extracted words to form these new and previously unseen summary sentences, which we refer to as *Non-Verbatim Sentences*. Additionally, when we use such mechanisms for paraphrase generation and summarisation, we want to be assured that the generated summary accurately reflects the content of the source text from which it was generated.

For the purposes of easily porting the approach to various domains, we employ a statistical approach to text generation. We follow [Witbrock and Mittal, 1999] in using the Viterbi algorithm [Forney, 1973] to search through the sea of possible word sequences for summary sentences. We have developed an extension to this work that narrows the search space not only to those sequences that maintains a semblance of grammaticality (via *n*-grams), but further still to those that preserve dependency structure information found in the source material. This mechanism should result in improved grammaticality. Examples of generated sentences are shown in Figure 1.

As an example, consider the following ungrammatical word sequence typical of that produced by an bi-gram generator: *The relief workers distributed food shortages are an issue.* One explanation for the error is that the bigram *food shortages* occurred in a context where the word *food* is a modifier of the noun head *shortages*. However, an approach reliant on just bigrams would not be able to detect that the word *food* already has a governing head word, in this case *distributed*. Ideally, we want to avoid such sequences since they will result in text fragments, in this case *shortages are an issue*. We could, for example, record the fact that *food* is already governed by a verb and thus avoid appending a second governing word.

In addition to *n*-grams, the extension propagates dependency features to constrain the search space. A dependency representation of the partially generated sentence is used to influence the choice of the next word in the sentence. This representation is a simplified form of a shallow syntactic dependency structure [Kittredge and Mel'cuk, 1983] which has had dependency role labels discarded. To obtain this structure, a statistical (dependency-based) parsing mechanism is folded into the Viterbi algorithm, producing a dependency structure for each search path that is followed.

We think of dependency structure here as a poor but easily obtainable surrogate for a semantic representation. Though not ideal as a semantic representation (for example, it still contains auxiliary verbs and other surface syntactic features), this tree structure can be seen as encoding predicate-argument structure. Thus, as a by-product of the grammaticality mechanism in our extension, we hope that this will also improve the chances that the generated sentence reflects the content of the source text.

Ideally, we would determine if the generated sentence is entailed by the source sentences in order to gauge if it accurately reflects its content. However, the entailment mechanisms that might provide such an answer are always only as good as the knowledge bases available to perform the necessary inferences. Unfortunately, good knowledge bases are often hard to obtain.

Instead, a preliminary evaluation measures the recall and precision of dependency relations as a crude approximation to entailment. This evaluation says nothing about whether the generated sentence is true or not. For example, the omission of an adverbial relation (for example, negation) could have dire consequences on the truth value of the generated sentence.

Nevertheless, we can use this measurement to distinguish between grammatical but unrelated sentences and those that are related but whose truth value is unknown. That is, if this generated dependency structure replicates (if even partially) a reference dependency structure from a source sentence, we assume that this indicates (partial) evidence supporting the relatedness of the propositional content of the generated sentence. We call this set of sentences *Verisimilitudes*.

In the remainder of this paper, we discuss sentence generation as a search process in Section 2. Section 3 provides a detailed account of our extension to the Viterbi algorithm which considers a statistical model of dependency relations. In Section 4, we examine related work in the summarisation and paraphrase community. The evaluation and results are discussed in Section 5. Finally, we conclude with future work in Section 6.

2 Narrowing the Search Space: A Description of the Statistical Sentence Generation Problem

In this work, sentence generation is couched as a search for the most probable sequence of words, given the vocabulary of some source text. However, this constitutes an enormous space which requires efficient searching. Whilst reducing a vocabulary to a suitable subset narrows this space somewhat, we can use statistical models, representing properties of language, to further prune the search space of word sequences to those strings that reflect real language usage. For example, *n*-gram models limit the word sequences examined to those that seem grammatically correct, at least for small windows of text.

However, *n*-grams alone often result in sentences that, whilst near-grammatical, are often just gibberish. When combined with a (word) content selection model, we narrow the search space even further to those sentences that appear to make sense. Accordingly, approaches such as Witbrock and Mittal [1999] and Wan et al. [2003] have investigated models that improve the choice of words in the sentence. Witbrock and Mittal's content model chooses words that make good headlines, whilst that of Wan et al. attempts to ensure that, given a short document like a news article, only words from sentences of the same subtopic are combined to form a new sentences. In this paper, we narrow the search space to those sequences that conserve dependency structures from within the input text.

Original Text
A military transporter was scheduled to take off in the afternoon from Yokota air base on the outskirts of Tokyo and fly to Osaka with 37,000 blankets .

Mondale said the United States, which has been flying in blankets and is sending a team of quake relief experts, was prepared to do more if Japan requested .

United States forces based in Japan will take blankets to help earthquake survivors Thursday, in the U.S. military's first disaster relief operation in Japan since it set up bases here.

Our approach with Dependencies
6: united states forces based in blankets
8: united states which has been flying in blankets
11: a military transporter was prepared to osaka with 37,000 blankets
18: mondale said the afternoon from yokota air base on the united states which has been flying in blankets
20: mondale said the outskirts of tokyo and is sending a military transporter was prepared to osaka with 37,000 blankets
23: united states forces based in the afternoon from yokota air base on the outskirts of tokyo and fly to osaka with 37,000 blankets
27: mondale said the afternoon from yokota air base on the outskirts of tokyo and is sending a military transporter was prepared to osaka with 37,000 blankets
29: united states which has been flying in the afternoon from yokota air base on the outskirts of tokyo and is sending a team of quake relief operation in blankets
31: united states which has been flying in the afternoon from yokota air base on the outskirts of tokyo and is sending a military transporter was prepared to osaka with 37,000 blankets
34: mondale said the afternoon from yokota air base on the united states which has been flying in the outskirts of tokyo and is sending a military transporter was prepared to osaka with 37,000 blankets
36: united states which has been flying in japan will take off in the afternoon from yokota air base on the outskirts of tokyo and is sending a military transporter was prepared to osaka with 37,000 blankets

Figure 1: A selection of example output. Sentences are prefixed by their length.

Our algorithm extension essentially passes along the long-distance context of dependency head information of the preceding word sequence, in order to influence the choice of the next word appended to the sentence. This dependency structure is constructed statistically by an O(n) algorithm, which is folded into the Viterbi algorithm. Thus, the extension is an O(n^4) algorithm. The use of dependency relations further constrains the search space. Competing paths through the search space are ranked taking into account the proposed dependency structures of the partially generated word sequences. Sentences with probable dependency structures are ranked higher. To model the probability of a dependency relation, we use statistical dependency models inspired by those described in Collins [1996].

3 A Mechanism for Propagating Dependency Features in the Extended Viterbi Algorithm

In this section, we present an overview of the main features of our algorithm extension. The Viterbi algorithm (for a comprehensive overview, see [Manning and Schütze, 1999]) is used to search for the best path across a network of nodes, where each node represents a word in the vocabulary. The best sentence is a string of words, each one emitted by the corresponding visited node on the path.

In this work, we begin with a Hidden Markov Model (HMM) where the nodes (ie, states) of the graph are uniquely

labelled with words from a relevant vocabulary. To obtain a suitable subset of the vocabulary, words are taken from a set of related sentences, such as those that might occur in a news article (as is the case for the original work by Witbrock and Mittal). We use the clusters of event related sentences from the Information Fusion work by Barzilay et al. [1999]. The edges between nodes in the HMM are typically weighted using bigram probabilities extracted from a related corpus.

Arcs between nodes are weighted using two pieces of information: a bigram probability corresponding to that pair of words; and a probability corresponding to the likelihood of a dependency relation between that pair of words. Both probabilities are computed using Maximum Likelihood Estimation.

In our extension, we modify the definition of the Transition Probability such that not only do we consider bigram probabilities but also dependency-based transition probabilities. Examining the dependency head of the preceding string allows us to consider long-distance context when appending a new word. The algorithm ranks highly those words with a plausible dependency relation to the preceding string, given the source text. To combine both the bigram and dependency models, we take the average of the dependency transition probability and the bigram probability.

To test only the effect of the dependency and *n*-gram-based transition probability in this evaluation, we assume that the emission probability is always one. The emission probability is interpreted as being a *Content Selection* mechanism that chooses words that are likely to be in a summary. Thus, in this paper, each word has an equally likely chance of being selected for the sentence.

Transition Probability is defined as:

$$p_{tr}(w_{i+1}|w_i) = average(p_{tr_{ngram}}(w_{i+1}|w_i), p_{tr_{dep}}(w_{i+1}|w_i))$$

where

$$p_{tr_{ngram}}(w_{i+1}|w_i) = \frac{count(w_i, w_{i+1})}{count(w_i)}$$

Emission Probability (for this paper, always set to 1):

$$p_{em}(w) = 1$$

Path Probability is defined recursively as:

$$p_{path}(w_0, \ldots, w_{i+1}) = p_{tr_{ngram}}(w_{i+1}|w_i) \times p_{em}(w) \times p_{path}(w_0 \ldots w_i)$$

Thus, our extension has two components: *Dependency Transition* and *Head Stack Reduction*. Aside from these modifications, the Viterbi algorithm remains the same.

3.1 Preprocessing

When the system is given a new text to summarise, we first segment the text into sentences and then parse each sentence to obtain corresponding dependency parse trees. These trees are then used to populate two adjacency matrices. Because the status of a word as a head or modifier depends on the word order in English, we consider relative word positions to determine if a relation has a forward or backward[1] direction.

[1]These are defined analogously to similar concepts in Combinatory Categorial Grammar [Steedman, 2000].

The relief workers distributed food to the hungry.
The UN workers requested medicine and blankets.

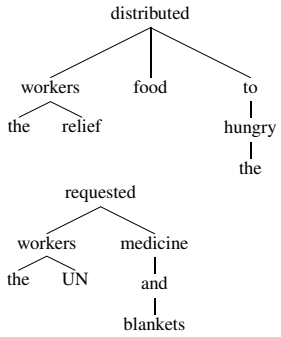

Figure 2: Two dependency trees from which the system will generate a sentence.

Forward and backward directional relations are stored in two separate matrices.

The first matrix, labelled Adj_{right}, stores (forward) relations in which the head of the relation is to the right of the modifier. The second matrix, Adj_{left}, stores (backward) relations in which the head is to the left of the modifier.

Presently, we only store dependency structures (both left and right) derived from the source text being summarised. One short-coming of this is that a dependency model built upon relationships in the source text will be very sparse. To allow the use of back-off mechanisms, we intend to keep track of a second set of adjacency matrices that would store dependency structures (again, both left and right) from a corpus. This is currently work in progress.

3.2 The Dependency Transition Probability

An Example Walkthrough

Given two input sentences *The relief workers distributed food to the hungry* and *The UN workers requested medicine and blankets*, and the corresponding dependency parses in Figure 2, the task is to generate a single sentence that contains material from these two sentences. As in [Barzilay *et al.*, 1999], we assume that the sentences stem from the same event, with the result that references can be fused together.

Imagine also that bigram frequencies have been collected from a relevant UN Humanitarian corpus. Figure 3 presents bigram probabilities and two sample paths through the lattice. The path could follow one of two forks after encountering the word *distributed*, since the corpus could have examples of the word pairs *distributed food* and *distributed blankets*. Since both *food* and *blankets* can reach the end-of-sentence state, both might conceivably be generated by considering just *n*-grams. However, only one is consistent with the input text.

To encourage the generation of verisimilitudes, we check for a dependency relation between *blankets* and *distributed* in the input sentence. As no evidence is found, we score this transition with a low weight. In contrast, there is evidence for the alternative path since the input text does contain a dependency relation between *food* and *distributed*.

Graph nodes:
w_1 is *workers*
w_2 is *distributed*
w_3 is *food*
w_4 is *blankets*
e is the *end-of-sentence* state

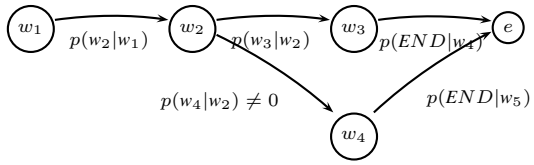

Figure 3: Two search paths. One is consistent with the input text, the other is not. Assume that the probabilities are taken from a relevant corpus such that $p(blankets|distributed)$ is not zero.

In reality, multiple words might still conceivably be modified by future words, not just the immediately preceding word. In this example, *distributed* is the root of a dependency tree structure representing the preceding string. However, any node along the rightmost root-to-leaf branch of the dependency tree (that represents the partially generated string) could be modified. This dependency structure is determined statistically using a probabilistic model of dependency relations. To represent the rightmost branch, we use a stack data structure (referred to as the *head stack*) whereby older stack items correspond to nodes closer to the root of the dependency tree.

Computing the Dependency Transition Probability
The probability of the dependency-based transition is estimated as follows:

$$p_{tr_{dep}}(w_{i+1}|w_i) \approx p(Dep_{sym}(w_{i+1}, headStack(w_i)) = \max_{h \in headStack(w_i)} p(Dep_{sym}(w_{i+1}, h))$$

where Dep_{sym} is the symmetric relation stating that some dependency relation occurs between a word and any of the words in the stack, irrespective of which is the head.

The probability indicates how likely it is that the new word can attach itself to this incrementally built dependency tree, either as a modifier or a governing head of some node in the tree. Maximising this probability allows us to find the single best attachment point. Since the stack is cumulatively passed on at each point, we need only consider the stack stored at the preceding word.

The function $Dep_{sym}/2$ provides the probabilities of a dependency relation and accounts for the direction of the relation. For two words a and b where a precedes b in the generated string, we have

$$p(Dep_{sym}(a,b)) \approx \frac{Adj_{right}(a,b) + Adj_{left}(b,a)}{cnt(\text{co-occur}(a,b))}$$

where Adj_{right} and Adj_{left} are the right and left adjacency matrices taken from the source text. In these matrices, row indices are heads and column indices are modifiers. Items on the stack are annotated to indicate whether or not they already have a governing head. If the proposed dependency relation attempts to attach a new governing head for one which already has one, the probability is set to zero.

As mentioned above, to handle sparsity in the dependency model, we intend to use a back-off approach to smooth the dependency model. We envisage a linear combination of source text dependencies and corpus-based dependencies. We do note that incorporating any smoothing technique on a dependency model might improve grammaticality at the expense of conserving source text dependencies.

3.3 The Head Stack Representation

Maintaining Head Stack

Once we decide that a newly considered path is better than any other previously considered one, we update the head stack to represent the extended path. At any point in time, the stack represents the rightmost root-to-leaf branch of the dependency tree (for the generated sentence) that can still be modified or governed by concatenating new words to the string.[2] Within the stack, older words may be modified by newer words. Our rules for modifying the stack are designed to cater for a projective[3] dependency grammar.

There are three possible alternative outcomes of the reduction. The first is that the proposed top-of-stack (ToS) has no dependency relation to any of the existing stack items, in which case the stack remains unchanged. For the second and third cases, we check each item on the stack and keep a record only of the best probable dependency between the proposed ToS and the appropriate stack item. The second outcome, then, is that the proposed ToS is the head of some item on the stack. All items up to and including that stack item are popped off and the proposed ToS is pushed on. The third outcome is that it modifies some item on the stack. All stack items up to (but not including) the stack item are popped off and the proposed ToS is pushed on. The pseudocode is presented in Figure 4. An example of stack manipulation is presented in Figure 5.

Although our extension caters for a right branching language such as English, we expect that an analogous algorithm exists for left branching languages, such as Japanese.

The algorithm in Figure 4 relies on three external functions. The first function, $dep_{sym}/2$, has already been presented above. The $annotateHeadedness/1$ function simply records if the input parameter is the head of the proposed dependency relation. The function, $isReduced/2$, relies on an auxiliary function returning the probability of one word being governed by the other, given the relative order of the words. In essence, this is our parsing step, determining which word governs the other. The function is defined as follows:

$$isReduced(w_1, w_2) = p(isHeadRight(w_1, w_2)) > p(isHeadLeft(w_1, w_2))$$

[2]Note that we can scan through the stack as well as push onto and pop from the top; this is thus the same type of stack as used in, for example, Nested Stack Automata.

[3]That is, if w_i depends on w_j, all words in between w_i and w_j are also dependent on w_j.

```
reduceHeadStack(aNode, aStack) returns aStack
Node_new ←aNode
Stack ←aStack # duplicate
Node_max ←NULL
Edge_prob ←0

# Find best attachment point
While notEmpty(aStack)
  Head ←pop(aStack)
  if p(dep_sym (Node_new, Head)) > Edge_prob
    Node_max ←Head
    Edge_prob ←dep_sym(Node_new, Head)

# Remove subtree under attachment point
While top(aStack) ≠ Node_max
  pop(aStack)

# Determine new head of existing string
if isReduced(Node_new, Node_max)
  pop(aStack)

annotateHeadedness(Node_new)
push(Node_new, aStack)
```

Figure 4: Pseudocode for the Head Stack Reduction operation

where w_1 precedes w_2, and:

$$p(isHeadRight(w_1, w_2)) \approx \frac{Adj_{right}(w_1, w_2)}{cnt(hasRelation(w_1, w_2, wherei(w_1) < \mathrm{i}(w_2)))}$$

and similarly,

$$p(isHeadLeft(w_1, w_2)) \approx \frac{Adj_{left}(w_2, w_1)}{cnt(hasRelation(w_1, w_2, wherei(w_1) < \mathrm{i}(w_2)))}$$

where $hasRelation/2$ is the number of times we see the two words in a dependency relation, and where $i(w_i)$ returns a word position in the corpus sentence. The function $isReduced/2$ makes calls to $p(isHeadRight/2)$ and $p(isHeadLeft/2)$. It returns true if the first parameter is the head of the second, and false otherwise. In the comparison, the denominator is constant. We thus need only the numerator in these auxiliary functions. In future work, we will use the same back-off procedure as described above, consulting the source material first and then the corpus.

Collins' distance heuristics [1996] weight the probability of a dependency relation between two words based on the distance (measured by sentence position) between them. We plan to implement a similar strategy by favouring small reductions in the head stack in the future. Thus, a reduction with a more recent stack item which is closer to the proposed ToS would be less penalised than an older one.

An Example Walkthrough

We now step through the generation of the sentence *The UN relief workers distributed food to the hungry*. Figure 5 shows how the head stack update mechanism updates and propagates the stack of governing words as we append words to the path to produce this string.

We first append the determiner *the* to the new string and push it onto the empty stack. As dictated by a high *n*-gram probability, the word *UN* follows. However, there is no evidence of a relation with the preceding word, so we simply push it on the stack. Similarly, *relief* is appended and also pushed on the stack.

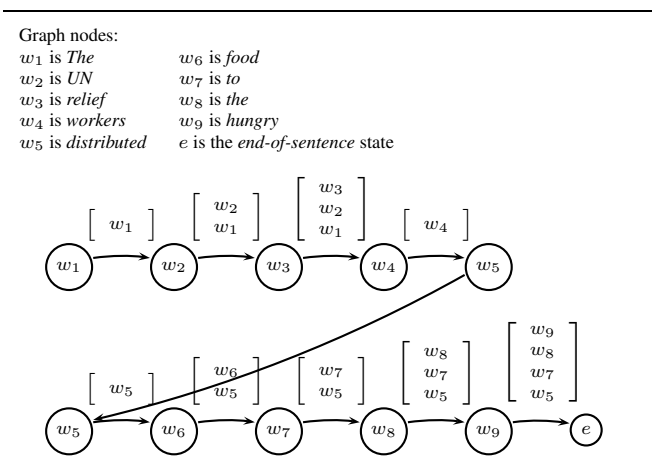

Figure 5: Propagating the head stack feature along the path.

When we encounter the word *workers* we find evidence that it governs each of the preceding three words. The modifiers are popped off and *workers* is pushed on. Skipping ahead, the transition *distribute food* has a high bigram probability and evidence for a dependency relation exists. This results in a strong overall path probability as opposed to the alternative fork in Figure 3. Since *distributed* can still be modified in the future by words, it is not popped off. The word *food* is pushed onto the stack as it too can still be modified.

The sentence could end there. Since we multiply path, transition and emission probabilities together, longer sentences will have a lower probability and will be penalised. However, we can choose to continue the generation process to produce a longer sentence. The word *to* modifies *distributed*. To prevent crossing dependencies, *food* is popped off the stack before pushing *to*. Appending the rest of the words is straightforward.

3.4 The Notion of Sentence-hood

Once a path reaches the *end-of-sentence* state, we can check the propagated head stack to see if it is a valid dependency structure.

Assuming that our *n*-gram model is accurate and the end-of-sentence marker is reached correctly, we can examine the head stack to see if the preceding string is acceptably structured. To do so, we simply perform a series of head stack reductions. Due to the way that we have defined the manipulation of the head stack so far, the only reductions possible here are cases where ToS is a modifier of an earlier stack object. Items are popped off the stack if this rule holds. After repeated reductions we should be left with a single word that represents the root of the dependency tree for the generated sentence. If this is the case, then the sentence is acceptable.

In practice, we set an upper bound on the sentence length, though the algorithm is free to terminate before then. The pseudocode to accept a sentence is presented in Figure 6. The extended Viterbi algorithm is presented in Figure 7.

```
isAcceptable(headStack) returns boolean
  while (headStack not empty)
    oldToS ←pop(headStack)
    if (not isHeadLeft(ToS,oldToS))
      return false
  if (isFinal(ToS))
    return true
  else
    return false
```

Figure 6: Pseudocode for accepting a generated sentence. The function $isFinal/1$ returns true if the word is suitable as the root of a dependency tree. For example, verbs and conjunctions would return true.

```
viterbiSearch(maxLength, stateGraph) returns bestPath
  numStates ←getNumStates(stateGraph)
  viterbi ←a matrix[numStates+2,maxLength+2]
  viterbi[0,0].score ←1.0
  for each time step t from 0 to maxLength do

    # Termination Condition
    if ((viterbi[endState, t].score ≠ 0)
      AND isAcceptable(endState.headStack))
        # Backtrace from endState and return path

    # Continue appending words
    for each state s from 0 to numStates do
      for each transition s' from s
        newScore ←
          viterbi[s,t].score × p_tr(s'|s) × p_em(s')
        if ((viterbi[s',t+1].score = 0) OR
            (newScore > viterbi[s', t+1]))
          viterbi[s',t+1].score ←newScore
          viterbi[s',t+1].headStack ←
            reduceHeadStack(s',viterbi[s,t].headStack)
          backPointer[s',t+1] ←s

  Backtrace from viterbi[endState,t] and return path
```

Figure 7: Extended Viterbi Algorithm

4 Related Work

There is a wealth of relevant research related to sentence generation. We focus here on a discussion of related work from statistical sentence generation and from summarisation.

In recent years, there has been a steady stream of research in statistical text generation. We will briefly discuss work which generates sentences from some sentential semantic representation via a statistical method. For examples of related statistical sentence generators see Langkilde and Knight [1998] and Bangalore and Rambow [2000]. These approaches begin with a representation of sentence semantics that closely resembles that of a dependency tree. This semantic representation is turned into a word lattice. By ranking all traversals of this lattice using an *n*-gram model, the best surface realisation of the semantic representation is chosen. The system then searches for the best path through this lattice. Our approach differs in that we do not start with a semantic representation. Instead, we paraphrase the original text. We search for the best word sequence and dependency tree structure concurrently.

Research in summarisation has also addressed the problem of generating non-verbatim sentences; see [Jing and McKeown, 1999], [Barzilay *et al.*, 1999] and more recently [Daumé III and Marcu, 2004]. Jing presented a HMM for learning alignments between summary and source sentences trained using examples of summary sentences generated by humans. Daume III also provides a mechanism for subsentential alignment but allows for alignments between multiple sentences. Both these approaches provide models for later recombining sentence fragments. Our work differs primarily in granularity. Using words as a basic unit potentially offers greater flexibility in pseudo-paraphrase generation since we able to modify the word sequence within the phrase.

It should be noted, however, that a successful execution of our algorithm is likely to conserve constituent structure (ie. a coarser granularity) via the use of dependencies, whilst still making available a flexibility at the word level. Additionally, our use of dependencies allows us to generate not only a string but a dependency tree for that sentence.

Finally, instance-based generation mechanisms have been proposed by Varges [Varges, 2003] in which cosine similarity metrics are used to find the closest training instance to the current sentence being generated. The generation mechanism proceeds to minimise the distance from this training instance. Whereas Varges would use a grammar for generation, we produce our strings using *n*-grams.

5 Evaluation

In this section, we describe a preliminary experiment designed to evaluate whether a dependency-based statistical generator improves content overlap. We use a precision and recall styled metric on dependency relations. We find that our approach performs significantly better than the bigram baseline.

In the interests of minimising conflating factors in this comparison, we simplify the problem by building a bigram language model for each input cluster of text. This provides

both the bigram baseline and our system with the best possible chance of producing a grammatical sentence with the vocabulary of the input text. Note that the baseline is a difficult one to beat because it is likely to reproduce long sequences from the original sentences. However, an exact regurgitation of input sentences is not necessarily the outcome of the baseline generator since, for each cluster, the model is built from multiple sentences.

We do not use any smoothing algorithms for dependency counts in this evaluation since we do not back-off to corpora-based dependency probabilities at present time. Thus, given the sparseness arising from a small set of sentences, our dependency probabilities tend towards boolean values. For both our approach and the baseline, the bigrams are smoothed using Katz's back-off method.

The data for our evaluation cases is taken from the information fusion data collected by [Barzilay *et al.*, 1999]. This data is made up of news articles that have first been grouped by topic, and then component topic sentences further clustered by similarity of event. We use 100 sentence clusters and on average there are 4 sentences per cluster. Each sentence in a cluster is parsed using the Connexor dependency parser (www.connexor.com) to obtain dependency relations.

Each sentence cluster forms an evaluation case in which we generate a single sentence. For each evaluation case, the baseline method and our method generates a set of answer strings, from 1 to 40 words in length. The average sentence length is about 30 words. To minimise the number of confounding factors, we turned off the sentence-hood check in our system since it is the modified transition probability and head stack reduction mechanisms that we want to test.

The generated strings of the baseline and our system were then parsed by the Connexor parser. We compared the output dependency relations against those of the input cluster using a precision and recall metric.
The precision metric is as follows:

$$\text{precision} = \frac{\text{count(matched-relations)}}{\text{count(generated-relations)}}$$

The corresponding recall metric is defined as:

$$\text{recall} = \frac{\text{count(matched-relations)}}{\text{count(source-text-relations)}}$$

5.1 Results and Discussion

Figure 8 shows the average precision for each generated sentence length across all test cases. The results show that our dependency-based system scored better than a bigram baseline.

Using a two-tailed Wilcoxon test (alpha = 0.05), we found that the differences in precision scores are significant for most sentence lengths except lengths 17 and 32. The failure to reject the null hypothesis[4] for these lengths is interpreted as idiosyncratic in our data set. Ignoring the two outliers, we

[4]That is, the means of scores by our system and the baseline are not different.

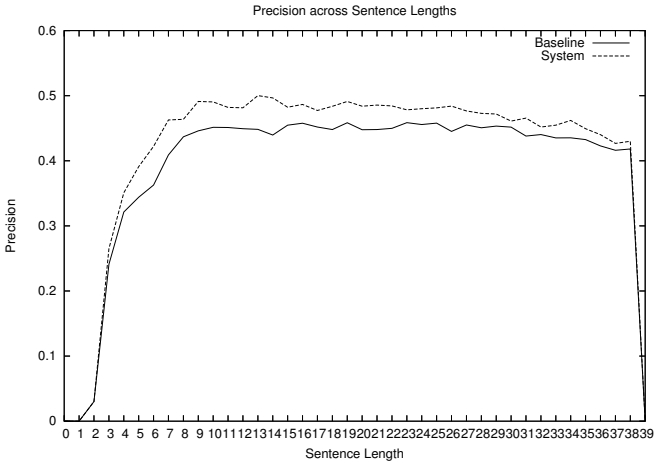

Figure 8: Dependency relation precision scores for generated output

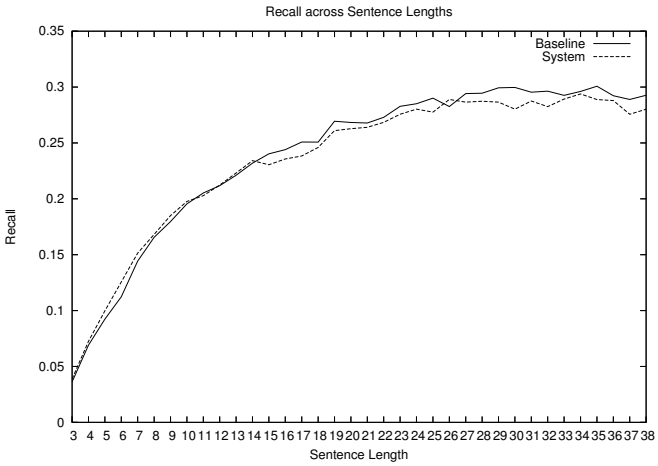

Figure 9: Dependency relation recall scores for generated output

reject the null hypothesis in general and conclude that using a dependency model in addition to a bigram model does improve precision.

The recall scores are shown in Figure 9. Unsurprisingly, there appears to be little difference between our approach and the baseline. Differences were not significant using the Wilcoxon test. This is consistent with the fact that our approach was designed to improve the content overlap and grammaticality of the generated sentence. As grammaticality was the key focus of the extension, there is no reason to believe that it should improve on the number of dependency relations recalled.

6 Conclusion and Future Work

In this paper, we have described our extension to the Viterbi algorithm that allows its use as a statistical generator that incorporates a dependency model. Sentences generated by this approach utilise dependency structures from the source text and are more grammatical than a bigram baseline. In addition, the mechanism also improves the degree of content overlap with the source text. In future work, we intend to explore the use of *n*-gram and dependency models built from a larger corpus in order to study their effects on grammaticality and content preservation. We also intend to compare our approach with the Information Fusion work of Barzilay [Barzilay *et al.*, 1999]. Finally, we simplified the content selection model in order to test the generation mechanisms. In future work, we intend to integrate previous content selection models with our dependency-based approach.

7 Acknowledgements

This work was funded by the Centre for Language Technology at Macquarie University and the CSIRO Information and Communication Technology Centre. We would like to thank the research groups of both organisations for useful comments and feedback.

References

[Bangalore and Rambow, 2000] Srinivas Bangalore and Owen Rambow. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th Conference on Computational Linguistics (COLING'2000), July 31 - August 4 2000*, Universität des Saarlandes, Saarbrücken, Germany, 2000.

[Barzilay *et al.*, 1999] Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th conference on Association for Computational Linguistics*, pages 550–557, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[Collins, 1996] Michael John Collins. A new statistical parser based on bigram lexical dependencies. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 184–191, San Francisco, 1996. Morgan Kaufmann Publishers.

[Daumé III and Marcu, 2004] Hal Daumé III and Daniel Marcu. A phrase-based hmm approach to document/abstract alignment. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 119–126, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[Forney, 1973] G. David Forney. The viterbi algorithm. *Proceedings of The IEEE*, 61(3):268–278, 1973.

[Jing and McKeown, 1999] Hongyan Jing and Kathleen McKeown. The decomposition of human-written summary sentences. In *Research and Development in Information Retrieval*, pages 129–136, 1999.

[Kittredge and Mel'cuk, 1983] Richard I. Kittredge and Igor Mel'cuk. Towards a computable model of meaning-text relations within a natural sublanguage. In *IJCAI*, pages 657–659, 1983.

[Knight and Marcu, 2002] Kevin Knight and Daniel Marcu. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artif. Intell.*, 139(1):91–107, 2002.

[Langkilde and Knight, 1998] Irene Langkilde and Kevin Knight. The practical value of N-grams in derivation. In Eduard Hovy, editor, *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 248–255, New Brunswick, New Jersey, 1998. Association for Computational Linguistics.

[Manning and Schütze, 1999] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[Steedman, 2000] Mark Steedman. *The syntactic process*. MIT Press, Cambridge, MA, USA, 2000.

[Varges, 2003] Sebastian Varges. *Instance-based Natural Language Generation*. PhD thesis, Institute for Communicating and Collaborative Systems, School of Informatics, University of Edinburgh, 2003.

[Wan *et al.*, 2003] Stephen Wan, Mark Dras, Cecile Paris, and Robert Dale. Using thematic information in statistical headline generation. In *The Proceedings of the Workshop on Multilingual Summarization and Question Answering at ACL 2003*, Sapporo, Japan, July 2003.

[Witbrock and Mittal, 1999] Michael J. Witbrock and Vibhu O. Mittal. Ultra-summarization (poster abstract): a statistical approach to generating highly condensed non-extractive summaries. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 315–316, New York, NY, USA, 1999. ACM Press.

Deriving content selection rules from a corpus of non-naturally occurring documents for a novel NLG application

Sandra Williams and Ehud Reiter
University of Aberdeen
Computing Science Department, Aberdeen, AB24 3UE, U.K.
{ swilliam, ereiter }@csd.abdn.ac.uk

Abstract

We describe a methodology for deriving content selection rules for NLG applications that aim to replace oral communications from human experts by written communications that are generated automatically. We argue for greater involvement of users and for a strategy for handling sparse data.

1 Introduction

One of the challenges in building NLG systems is to derive content selection (CS) rules, typically by combining corpus knowledge and domain (and user) knowledge. CS rules specify what information the output document should communicate for a particular set of input data. For instance, they often map input data to fragments of document content.

Novel Natural Language Generation (NLG) applications produce documents that do not occur naturally. A subset of these applications communicate information *in written form* that humans would normally deliver *orally*. Such applications pose particular problems for NLG developers as we demonstrate in this paper by using our own development of the SkillSum system as a case study. More specifically we describe the part of SkillSum's development that involved the derivation of content selection (CS) rules.

SkillSum [Williams and Reiter, 2005] generates personalised basic skills feedback reports. Basic-skills tutors usually give feedback orally, but SkillSum attempts to generate a written report that communicates similar information to an adult student who has just completed a basic skills test. STOP [Reiter *et al*., 2003b] was another NLG system that attempted to communicate information that was normally delivered orally. STOP generated personalised smoking cessation letters, which communicated the kind of advice that would normally be given orally by a GP during a personal one-to-one consultation.

Tutors and doctors give their feedback orally partially because information about skills and health can be very sensitive and personal, and partially because discussing the topic orally is quicker (and hence cheaper) than writing a written report. They also recognize that there are advantages to written texts (notably that the recipient can take a written text home and think about it, and also discuss it with family and friends), and in general would like to have the option of giving people written texts, if these could be produced quickly and cheaply. Hence our interest to using NLG to automatically produce such texts.

The problem of deriving CS rules from corpora and domain/user knowledge is very hard, but is often glossed over; it is assumed that we can magically come up with successful CS rules even when there is little evidence for what "good content" might be for a given application or whether the content will be right for users. We can derive CS rules from corpora if we have a large parallel corpus of input data and manually-authored output texts, which covers most permutations of inputs and outputs (e.g. the parallel corpus of the SumTime system [Sripada *et al*. 2003]). Another situation where it might be relatively easy to derive CS rules is if the specification of what content should be present in the output is well defined and the application requires only a small, simple set of content and message types to be generated. In such cases, the methodologies presented by Reiter and Dale (2000) and Geldof (2003) can be used. However, none of these was the case with SkillSum (or in STOP).

In this paper we describe how a corpus of expert-authored basic skills reports was collected and analysed. We focus on the problem of CS rule derivation and in particular on problems we encountered with variability of content and sparsity of data in the corpus and in trying to incorporate the requirements of users which sometimes conflicted with experts' advice.

2 Related Work

Our starting point for determination of content selection rules was Reiter and Dale [2000], Geldof [2003] and Reiter and Sripada [2002]. Reiter and Dale describe a method for deriving knowledge from a corpus and Geldof essentially extends the method for message types, referring expressions, aggregation types and lexical choice. We followed Geldof's method to derive high-level document structure, but at lower-levels, the content of SkillSum's feedback reports is far less clear-cut than that of Geldof's route descriptions. There are fewer obligatory elements and there is not the same rigid logical ordering as that imposed by physical routes. We also incorporated many of the KA techniques discussed by Reiter *et al*. [2003a]. These methods work best when there are a few expert-authored texts for every possi-

ble set of system inputs. Since we did not have this, we extended the methodology to include extrapolation of rules to cover missing corpus data and allowed users to have a say in content selection. This follows Schneiderman's advice [2000] to accommodate users, even though they are not normally involved at the content selection stage.

SkillSum differs from existing NLG applications in that the content of its output texts serves very different communicative purposes from those of many existing NLG applications. In fact basic skills reports include multiple kinds of communicative purpose. Other systems generate descriptions, or explanations, or instructions, or advice. But SkillSum's content is complex in that it includes many of these: descriptions, interpretations, advice *and* instructions. SkillSum's content is also different from tutoring systems that generate explanations and instructions (e.g. [Moore *et al.*, 2004]) and from Cogentex's Recommender system that generates camera purchasing advice (see www.cogentex.com/solutions/recommender/). From this perspective it is perhaps most similar to STOP [Reiter *et al.*, 2003b].

A related text type to basic skills summary reports is school reports. However, Education literature on writing school reports is unhelpful because adult basic learners have often had bad experiences with school and school reports in the past [FENTO, 2004].

3 The SkillSum Application

SkillSum is a web-based application for basic skills testing and feedback report generation. Users of SkillSum can test either their literacy or numeracy in a short test consisting of a maximum of 27 questions and then receive a personalised report automatically generated using NLG technology (see Figure 1 for example output).

The intended users of SkillSum are adults aged 16 years and over with low basic skills, but not with severe learning difficulties. Such adults occur in large numbers in the UK; up to one fifth of the adult population, according to a Government survey [Moser, 1999]. SkillSum is a collaborative project between a commercial partner, Cambridge Training and Development Ltd. (CTAD) and researchers at Aberdeen University. CTAD developed SkillSum's basic skills testing module and Aberdeen the feedback report generator.

SkillSum's basic skills testing component originally contained much longer tests from which diagnostics were possible. Trials with users revealed that these tests would take too long to complete and users would need support whilst doing them. We switched to shorter tests that could be completed with little support. The shorter tests are "screeners", that is, they identify problems with basic skills but do not include diagnostics. This switch had quite an impact on feedback report content. For instance, the diagnostic part of reports became more general and vague.

Initially SkillSum was to be used at home or in Internet access centres. Now SkillSum is to be used in further education colleges where all incoming students are normally screened for basic skills problems so that help with basic English and Maths can be provided. Again, the change had an impact on feedback report content. Now users are college students who have just (or are just about to) enrol in a course. We hypothesised that such students would want to know if their skills were adequate for their intended course. Trials with students confirmed that this type of content was relevant.

We have run trials of SkillSum prototypes in a number of colleges. To date, we have developed five SkillSum prototypes and have carried out trials of each one in a continuous cycle of system development, testing, user trials and improvement.

4 Deriving CS Rules for SkillSum

In SkillSum, since the output texts did not occur naturally, we asked human experts to write some examples for us. This produced a fairly small corpus (Section 4.3), which suffered from data sparsity (i.e. corpus texts covered only a small fraction of the possible permutations of inputs to the system). We therefore found it necessary to manually extrapolate our CS rules to account for cases not covered by the corpus.

Reiter and Sripada (2002) found that corpora of expert-written texts cannot always be considered as gold standards for NLG because experts disagree and they can make mistakes. In SkillSum too, it was clear that our expert-authored corpus of sample basic skills feedback reports could not necessarily be regarded as the last word on what the content should be because our experts differed in their choice of content and users disagreed with experts.

We encountered another problem with SkillSum. Because our application was novel, we did not have a clear specification of what content should be included at the start of the project. In fact our specification developed gradually over the first year of the project. Other novel NLG applications may also suffer from this problem.

Reiter and Dale (2000) say that "*the goal of many NLG systems is to produce documents which are as similar as possible to documents produced by human experts*". Our goal was not just to mimic the content chosen by experts, but also to consider the requirements of users and to make the content useful and relevant to them. We therefore listened to users too and came up with CS rules for content that we hoped would be relevant and motivating for them as well as being acceptable to basic skills experts. Allowing users to have their say in the final specification of content is a departure from previous NLG CS rule derivation methodologies.

Our methodology was to work closely with both domain experts and with the intended users of the system and to combine knowledge acquired from them with knowledge from the corpus. This paper describes how we derived CS rules from a semi-automatically analysed corpus of human authored example output texts. Indeed, in some cases, as we will demonstrate in this paper, user requirements and extrapolation of rules played a more important role than corpus data.

In some NLG applications, if the content is wrong the cost might not be all that high. For example, irrelevant in-

formation could simply be ignored by users. But in SkillSum if the content is wrong, the cost is high; users can get angry, and (even worse) they may lose interest in improving their skills. Hence our development methodology stresses regularly producing and evaluating prototype systems, and using feedback from real users to improve document content.

4.1 What is in a basic skills summary report?

SkillSum reports should help people understand their basic skills strengths and weaknesses and advise them (if necessary) on how to get help. Although experts agreed that this type of content should be present, it still posed a challenge as to what to say exactly, because the topic is a very sensitive one indeed. Telling people with low self-confidence that they have problems with their literacy and/or numeracy can be hurtful! As mentioned above, the cost of getting the content wrong could be very high indeed.

Some issues that we considered include:

- Should reports mention students' mistakes as well as their correct answers?
- Should reports congratulate students for doing well, when perhaps their performance was worse than normal, or, on the other hand, should they commiserate with students when perhaps their performance was better than normal?
- Should reports attempt to motivate users by referring to their ambitions (e.g. qualifications or career) and how much (if any) motivational content should be included?
- How much advice should be given?

Experts tend to agree that reports should be encouraging, focus on positive aspects and not mention mistakes. On the other hand, the users often wanted to know what their mistakes were (it can be frustrating to score twenty-six out of twenty-seven and not know which one was wrong!). To address this, current reports include "more information" links to pop up a list of questions with correct and incorrect answers.

Not knowing an individual and how much effort he/she has put into the test remains an unsolved problem. Inclusion of evaluative comments such as *"this is very good"* is meaningless without such knowledge and, indeed they are meaningless without reference to some scale (but experts advised against mentioning the U.K. basic skills core curriculum scale as students are not familiar with it).

Another difficulty was what to say to people who could not answer many, or any, of the screener questions correctly. Also we cannot tell if this was because something went wrong during the test (since it is web-based): perhaps there was a problem with using the computer or the mouse; or perhaps the user has severe learning difficulties. Current reports for these users are very short indeed and advise talking to a tutor.

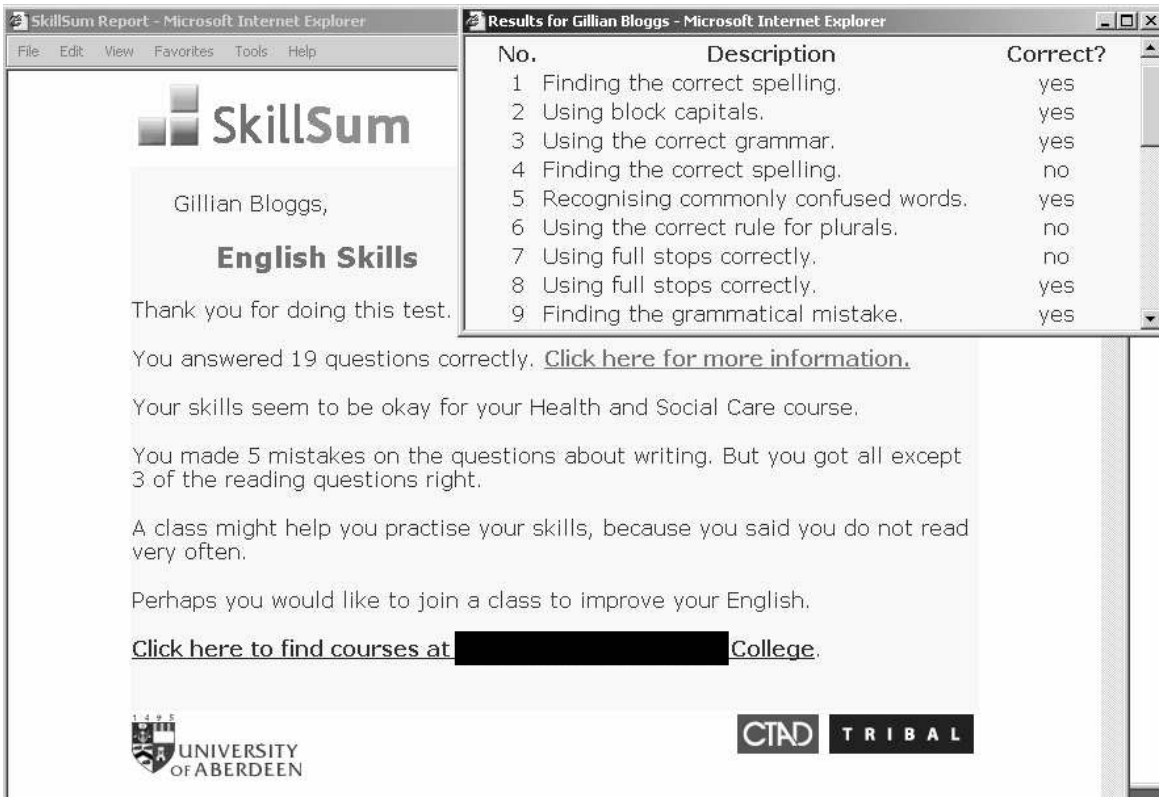

Figure 1. Output from SkillSum showing pop-up with "more information".

A student MSc project [Tintarev 2004], experimented with generating motivational content based on a user's intrinsic motivations (e.g. self-confidence and self-esteem) but an evaluation showed that it was not very successful because it is very difficult to obtain this data simply and reliably from users. This could have potential for future work but for the present, we confine SkillSum's motivational content to encouraging a user to gain a sufficient standard in basic skills for his/her course (since data about courses can be obtained simply and reliably).

4.2 Baseline System

CTAD's software currently generates very simple reports, which just give a score and overall literacy level, and suggest discussing this with a tutor. For example:

Andy Jones
Date: 06/10/2004
Thank you for doing this test. You scored 13. You may need help with level 1 literacy. Talk to your tutor or supervisor."

This is the baseline from which to measure NLG output.

4.3 KA and corpus collection

At the beginning of the project we collected a corpus of expert-authored texts relating to the longer basic skills test. For the new screener tests we elicited new expert-authored reports for nine case studies in literacy and nine in numeracy. An alternative method would have been to record tutor-student feedback sessions, but due to the sensitive and confidential nature of these, we preferred a method that was less intrusive; we also wanted the expert tutors to consider the issues involved in producing written reports. We gave them test results and short user profiles containing background material, e.g. age, current course and/or job wanted and self-assessments of maths and English skills (these were built using anonymous data from actual people who took part in earlier pilots and from [Swain *et al.*, 2004]). Two such profiles can be seen in the Appendix along with two example reports written by an expert.

We also acquired 1500 sets of test results (that is, input data for SkillSum); this gave us an idea of the range of inputs that SkillSum needed to cover.

4.4 Deriving high level document structure

An analysis of the tutor-authored reports demonstrated that they have similarities in high-level content structures but individual author differences in lower-level content. Our high-level analysis essentially followed the methodology of Geldof [2003]. Most of the expert-authored basic skills feedback reports included an initial section, a description of results (summary) an interpretation of the results (diagnosis), advice on what to do next (advice section) and a site-tailored link. For example, the report shown in Figure 1 has the following high-level structure.

- **Initial**:*"Gillian Bloggs, English Skills Thank you for doing this test."*
- **Summary**: *"You answered 19 questions correctly. Click here for more information."* Link to pop-up list of questions.
- **Diagnosis**: *"Your skills seem to be okay for your Health and Social Care course. You made 5 mistakes on the questions about writing. But you got all except 3 of the reading questions right."*
- **Advice**: *"A class might help you practise your skills, because you said you do not read very often. Perhaps you would like to join a class to improve your English."*
- **Site-tailored link**: *"Click here to find courses at Peterborough Regional College."*

Notice that even though Gillian's skills were considered good enough for her course, she is still being encouraged to take a class to improve. This is because her level of English is not very high.

Most expert-authored reports follow a similar basic structure. Sometimes "thanks" in the initial section was not present. The summary section was always present. The diagnosis section was not present in reports for students who had answered less than five questions correctly. As the overall score increased, the length of diagnosis and advice increased and sometimes these more complex sections were interleaved "diagnosis, advice, diagnosis, advice" and so on.

As the high-level structure emerged, it became clear what types of user knowledge we need to elicit in order to generate reports, e.g. self-assessments of skills, frequencies of reading, writing and calculations and information about users' courses.

4.5 Lower-level content

Lower-level content was analysed using a number of methods. We parsed the entire expert-authored corpus using MIT's parser (web.media.mit.edu/~hugo/montylingua/) and identified Message Types as Geldof suggests [2003]. The parsed corpus was used to model deep syntactic structures.

To derive rules for lower-level content, we attempted to draw flowcharts with decision points for individual messages inclusion, but we found the most successful method for SkillSum turned out to be manual RST analysis [Mann and Thompson 1998] of the entire corpus to give a number of RST tree fragment structures, e.g. one per piece of diagnosis or advice, like that shown in Figure 2.

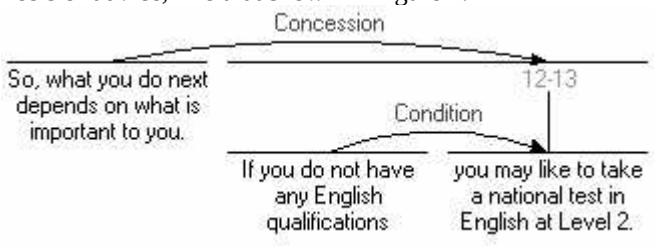

Figure 2. RST analysis of a piece of Advice (strings represent deep syntax).

This was followed by manual simplification (e.g. removal of cue phrases and concession satellite in Figure 2) and then construction of individual rules for inclusion of the templates (based on the user's results in the Screener and the user model). The document planning module outputs these RST structures which are then input to the microplanner that makes choices on ordering, cue phrases aggregation and punctuation, see Williams [2004].

4.6 Extrapolating CS rules to cover missing corpus data

The corpus was small so we extrapolated from existing data. In practice, CS rules are expressed as if-then rules and sometimes, for instance, the "else" part might be missing. Extrapolating the rule would mean supplying the missing part. For example, *"you should have the reading skills to be able to cope with your sports course"* was present (one occurrence) in the corpus, but there was no data about what to say when skills were inadequate for the user's course. The rule for including this content was therefore extrapolated by adding an "else" part to the rule as follows:

IF *the user is about to begin a Level 1 course at college*
AND *his/her English skills are at least Level 1,*
THEN
Add content to advise the user that his/her skills are adequate for his/her course.
ELSE,
Add content to advise the user that his/her skills are inadequate
IF *he/she is not already receiving help with basic skills,*
THEN *add content that he/she should try to improve his/her skills, e.g. by taking a basic skills course.*

Of course we piloted extrapolated rules with experts and users. The above rule turned out to be one of the most important CS rules in SkillSum (in the sense that it is currently deployed in the generation of every report where the user's course is known) although only part of it actually occurred in our corpus.

4.7 Incorporating domain knowledge and users' comments

The above rule incorporates domain knowledge about courses and levels, knowledge about the user (e.g. what course the user is about to take and whether he/she is receiving help with basic skills) and expert knowledge (advise the student to take up a basic skills course).

We also incorporated comments from users. For example, many users told us they wanted to know which questions they got right and wrong, so we added this information in a pop-up window (see Figure 1).

4.8 CS Rule Derivation Methodology in SkillSum

Current practice in the NLG community (at least as described in published papers) is to base CS rules on KA with experts and corpus analysis. The methodology we used in SkillSum also includes rule extrapolation and the involvement of end users. The following list summarises our method which we also illustrate in Figure 3.

Knowledge Acquisition:

- Interview experts to elicit domain knowledge.
- Collect sets of results and sets of personal details (user models).
- Ask experts to write example reports (corpus of output texts)

System Development:

- Derive rules for analysing existing input data.
- Determine what additional input data (if any) is needed (such as background information about the user).

.

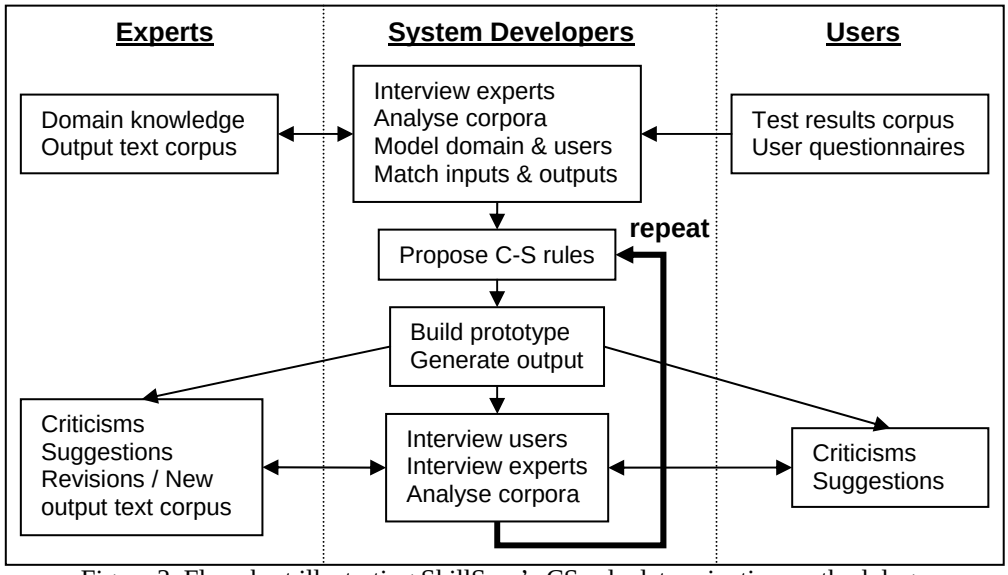

Figure 3. Flowchart illustrating SkillSum's CS rule determination methodology

- Derive content and message types from corpus texts, and use RST analysis.
- Derive rules for selecting content from the input data using available domain and user knowledge **and extrapolate**.

System Trial:

- Ask experts to criticise content of generated reports, edit them and write alternative reports.
- Pilot with users and ask them to criticise and suggest improvements.

The flowchart in Figure 3 shows how we incorporate knowledge from experts (on the left-hand side) and from users (on the right-hand side) when developing CS rules. The central system developer section shows how we repeat the last three stages a number of times. To date with SkillSum we have repeated these steps five times. At a recent trial of SkillSum, 13 out of 15 students preferred the content of the NLG output over that of the baseline system described in Section 4.2 (significant at $p<0.01$ in a binomial test).

5 Conclusions

CS rule derivation is a complex process. In this paper we have described our methodology for deriving CS rules for SkillSum, one of a class of novel NLG applications where output that is normally delivered orally by humans is generated by the NLG system in written form. We have argued that the methodology should include a strategy for handling sparse data when the corpus of expert-authored texts is small and also that users should have a chance say what they find useful and relevant in terms of document content. We feel that users can make a valuable contribution to the process of CS rule derivation. We hope that developers of similar NLG systems will find our experiences with SkillSum useful.

Acknowledgments

We thank the reviewers, CTAD and members of the Aberdeen NLG group for their help and comments. This research is supported by U.K. PACCIT-LINK Grant ESRC RES-328-25-0026.

References

[FENTO, 2004] Further Education National Training Organisation (FENTO). Including Language, Literacy and Numeracy Learning in all Post-16 Education. www.nrdc.org.uk.

[Geldof, 2003] Geldof, S. Corpus analysis for NLG. In: Reiter, E., Horacek, H. and van Deemter, K. (Eds.) *9th European Workshop on NLG*, 31-38.

[Mann and Thompson, 1987] W. Mann and S. Thompson. Rhetorical Structure Theory: A Theory of Text Organization. ISI, Reprint Series no. ISI/RS-87-190.

[Moore *et al.*, 2004] Johanna D. Moore, Kaska Porayska-Pomsta, Sebastian Varges, and Claus Zinn. Generating Tutorial Feedback with Affect. *Proc.17th International FLAIRS Conference*, AAAI Press.

[Moser 1999] Moser, C. Improving literacy and numeracy: a fresh start. Report of the working group chaired by Sir Clause Moser. www.lifelonglearning.co.uk/mosergroup/

[Reiter and Dale, 2000] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. CUP.

[Reiter and Sripada, 2002] Ehud Reiter and Somayajulu Sripada. Should Corpora Texts be Gold Standards for NLG? *Proceedings of INLG02*, 97-104.

[Reiter *et al*., 2003a] Ehud Reiter, Somayajulu Sripada and Roma Robertson. Acquiring Correct Knowledge for Natural Language Generation, *Journal of Artificial Intelligence Research*, 18:491-516.

[Reiter *et al.*, 2003b] Ehud Reiter, Roma Robertson, and Liesl Osman. Lessons from a Failure: Generating Tailored Smoking Cessation Letters. *Artificial Intelligence*, 44:41-58.

[Schneiderman, 2000] Ben Schneiderman. Universal Usability: Pushing human-computer interaction research to empower every citizen. Communications of the ACM, 43, 84-91.

[Sripada *et al*., 2003] S. G. Sripada, Ehud Reiter, Jim Hunter and Jin Yu. Exploiting a parallel TEXT-DATA corpus. *Proceedings of Corpus Linguistics*, 734-743.

[Swain *et al.*, 2004] Swain, Jon., Elizabeth Baker, Debbie Holder, Barbara Newmarch and Diana Coben. Beyond the daily application: Making numeracy teaching meaningful to adult learners. NRDC. www.nrdc.org.uk.

[Tintarev, 2004] Nava Tintarev. Content Determination for Reports Aimed at Adult Literacy Learners. MSc Thesis, Uppsala Universitet.

[Williams and Reiter, 2005] Sandra Williams and Ehud Reiter. Generating readable texts for readers with low basic skills. To appear in *Proceedings of the 10th European Workshop on NLG*.

[Williams, 2004] Williams, Sandra, H. *Natural Language Generation of discourse relations for different reading levels.* PhD Thesis, University of Aberdeen, 2004.

Appendix 1. Extracts from Experts' Questionnaire

Instructions for authors

We would like you to write (or edit) a short feedback report for each of the people described in the following learner profiles. Please try to use everything in the profile that you consider relevant. You can write anything you think appropriate, as if the people in the profiles had come to you personally as a consultant with their screener results. They are asking for feedback and advice and you want to encourage them as much as possible. Assume that you know everything that the profile says about them. It might help to imagine the person is sitting next to you and that you are talking to him/her.

What to include

- The report should tell the person how well they have done in the screener. We have given you their screener results and a list of the questions in each screener.
- If you think it is relevant, you could mention right or wrong answers to particular questions and/or particular skills that you think that the person has mastered, or has not yet been taught. How would you explain to this person what he/she is good at and/or what he/she still needs to learn?
- You want to encourage the person to improve his/her skills. Can you write something that he/she would find motivating?
- Please also tell us which parts of the "character and background" are useful.

Restrictions

- Reports for people with the lowest scores should be very short indeed (say, 50 words), but they still ought to contain some comments and advice that the person would find useful. Reports for people with higher scores can be a little longer.
- A recent CTAD strategy meeting decided, I think, that reports should not mention levels or the curriculum, so you ought not to mention those.

Learner Profile #2 (to be edited): Literacy Screener

Name: John Smith **Age:** 18 **His own assessment of his Reading and Writing:** "okay"
Overall Screener Score: 8 **Time taken**: 8 mins.

Background (Stoke-on-Trent)

"John is quite shy. His main motivation for studying is to get a better job. He wants to be a bricklayer or joiner. He believes that maths is important for this, but is not so sure about English. One reason for enrolling was seeing a friend who had done the same course get a job. He was disappointed at his performance in the test."

Results

Q. no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
status	r	r	r	w	w	r	w	r	w	r	w	w	r	w	r	w	w	w	w
entered	0	3	2	2	0	2	2	1	2	1	2	0	2	1	2	1	2	0	1

Q. no.	20	21	22	23	24	25	26	27
status	w	u	u	u	u	u	u	u
entered	25							

Report (to be edited)

John Smith,

English Skills

You got eight right. You were best at:

- grammar
- and reading to find things out.

Your friend got a job after doing the course and so can you. If you work hard at English,

- you will find it easier to write job applications
- and you will be more confident in job interviews.

CLICK HERE FOR SOME ENGLISH TIPS

Expert-authored report for John Smith

John got:

- *3 out of 3 Entry 3 questions right*
- *5 out of 12 Level 1 questions right (saw all of them)*
- *0 out of 12 Level 2 questions right (saw 5 of them)*

As he got 5 consecutive questions wrong (16 to 20) he only saw 20 questions in total and this took her 8 minutes, i.e. 24 seconds each question.

One issue for John is that he expressed disappointment at his results. I wonder what he was basing this feeling on. Does he already know what his results mean or did he realise that he was making errors?

Things I might say if he was with me are:

1. *Thanks for doing this.*
2. *Why are you disappointed?*
3. *You answered eight questions correctly.*
4. *What do you feel are your strong points?*
5. *Are there any areas you can think of where you might need some help?*
6. *From this short test I would suggest that you have a few gaps at Level 1. You do seem to be pretty good at finding out what you need to know when you read. Do you agree?*
7. *Would you like to join a class to fill these gaps?*
8. *If you do join a class the teacher will make sure that you can concentrate on what you need to learn. But it would be a good idea to get a little bit more information first on where those gaps are. I would like you to answer some more questions on another day. Are you happy to do that? With the information from that and from what you can tell us we should make sure that you plug those gaps.*
9. *Do you want to ask me anything about the questions or about what happens next?*

Learner Profile #7 (to be written from scratch): Literacy Screener

Name: Diana Hill **Age:** 16 **Her own assessment of her English:** ?

Overall Screener Score: 15 **Time taken:** 16 mins.

Background:

"Diana said that better reading skills would improve her self-confidence and also help her job-wise. She wants to work with children and wants to be able to read stories to them. She did poorly on the reading test, but said she liked to read. She thought she was okay at maths."

Results

Q. No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
status	r	r	w	w	r	r	w	w	r	r	r	r	w	r	r	w	w	r	w
entered	0	3	0	3	2	1	1	2	6	1	0	3	1	3	2	1	1	3	1

Q. No.	20	21	22	23	24	25	26	27
status	w	r	r	w	r	w	r	w
entered	15	1	3	1	0	0	1	3

Expert-authored report for Diana Hill

Diana got:

- *2 out of 3 Entry 3 questions right*
- *8 out of 12 Level 1 questions right (saw all of them)*
- *5 out of 12 Level 2 questions right (saw all of them)*

This took her 20 minutes, i.e. 35 seconds each question.

Things I might say if she was with me (and she was not already attending a class) are:

1. *Thanks for doing this.*
2. *You answered 15 questions correctly.*
3. *You only made mistakes on a couple of questions where you had to read. You said that you like reading - so that does not seem to be a problem for you at all. Do you agree?*
4. *The mistakes you did make were more to do with writing. It may be that you would like to improve your spelling and punctuation.*
5. *What you do next depends on what is important to you. If you do not have any English qualifications you may like to prepare for the national test in English at Level 1.*
6. *Do you want to ask me anything about the questions or about what happens next?*

Building Surface Realizers Automatically from Corpora *

Huayan Zhong and Amanda J. Stent
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400 USA
{huayan, stent}@cs.sunysb.edu

Abstract

In this paper, we evaluate the feasibility of automatically acquiring surface realizers from corpora using general-purpose parsing tools and lexicons. We present a basic architecture for acquiring a generation grammar, describe a surface realizer that uses grammars developed in this way, and present a set of experiments on different corpora that highlight possible improvements in our approach.

1 Introduction

The purpose of surface realization (SR) is to generate sentences or phrases from input semantic or syntactic representations that: a) preserve the input meaning; and b) are grammatical. To this end, it may select content words, insert function words, arrange words in order, and perform morphological inflection.

We are interested in modeling aspects of lexical and syntactic variation and lexical/synactic adaptation directly in the surface realizer (cf. [Creswell, 2003; Purver and Kempson, 2004]). It will be easier to do this if our surface realizer is directly acquired from data. In this paper, we explore the possibilities of automatically acquiring a domain-specific surface realizer from unlabeled data using general-purpose parsing tools and lexicons. We use our surface realizer in spoken and multimodal dialog systems with a variety of architectures, so we chose as our input format logical forms in which content words can, but need not be, specified. For this type of application, we are also concerned about speed of generation, so we looked at the possibility of eliminating a second-stage ranker.

In Section 2 we discuss previous approaches to surface realization. We describe our data in Section 3. In Section 4 we describe our approach to acquiring a generation grammar and our surface realizer. In Sections 6 and 5 we present the results of evaluations of several automatically-acquired generation grammars, and highlight areas for improvement.

*We would like to thank the blind reviewers for their helpful comments, and Gregory Aist and colleagues for collecting and transcribing part of the CALO corpus. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010.

2 Related Work

Most existing surface realizers use hand-written grammars or templates. FUF/SURGE [Elhadad, 1993; Elhadad and Robin, 1997] uses a complex unification grammar to generate sentences. The sentences it generates are of high quality. However, it can take quite a long time to generate a sentence [Callaway, 2003]. Genesis-II [Baptist and Seneff, 2000] and YAG [Channarukul, 1999] use hand-written templates in the air travel domain. Template-based surface realizers work faster than grammar-based surface realizers. However, they are domain-specific so must be changed for each new domain.

Two-stage surface realizers, e.g. [Langkilde, 2002; Oh and Rudnicky, 2002; Ratnaparkhi, 2000] use a hand-written grammar to generate potential outputs, and then a language model to rank these outputs. These surface realizers can produce output of quite high quality without the coding effort required to write a complex grammar by hand [Callaway, 2003].

Most statistical surface realizers are trained on annotated data (e.g. [Rambow and Bangalore, 2000; Marciniak and Strube, 2004; Pan and Shaw, 2004; S. Corston-Oliver and Moore, 2002]). FERGUS [Rambow and Bangalore, 2000] is a two-stage statistical surface realizer. It takes a dependency tree with content words specified as input, and realizes it by selection and combination of TAG derivation trees automatically acquired from an annotated corpus. It then generates a word lattice for the trees, and uses a language model to walk a highly probable path through the lattice, producing an output sentence. Marciniak and Strube report on a statistical surface realizer that generates from logical forms to output sentences [Marciniak and Strube, 2004]. This surface realizer consists of a set of eight classifiers trained on a set of 70 manually annotated texts.

Amalgam is a surface realizer trained on automatically parsed data and takes a logical form as input, so is perhaps closest to ours in style [S. Corston-Oliver and Moore, 2002]. It uses a classification-based approach. We are interested, however, in a direct representation of the mappings between semantics, syntax and realization so that we can manipulate these mappings on-the-fly to produce variation, so we use a probabilistic grammar instead. Our goal is to look at whether it is possible to create a framework in which domain-specific surface realizers can be acquired automatically from unannotated data using general-purpose tools and resources and then adapted during interaction.

Corpus	total # sents (unique sents)	total # words (unique words)
EXPL	5435 (5403)	86584 (9432)
CALO	4255 (2977)	65886 (4317)
WSJ	43411 (43162)	1044265 (40876)

Table 1: Size of corpora

3 Data

We used three corpora for these experiments: WSJ, EXPL and CALO (Table 1). The first, the Penn Wall Street Journal Treebank [Marcus *et al.*, 1993], we selected to allow us to compare performance with systems like FERGUS [Rambow and Bangalore, 2000] and HALogen [Langkilde, 2002], as well as to identify performance gains if there is "perfect" parsing. The second, EXPL, is a corpus of text explanations for children [howstuffworks, 2003], that we selected because of the relative simplicity of its sentence structures. Finally, we selected the CALO corpus, a collection of corpora of spoken dialog in purchasing domains, in order to allow us to look at performance gains if there are domain-specific lexicons and ontologies. Because of the small size of this corpus, we also report results for a grammer acquired from a combination of this corpus with the Penn Treebank. We randomly assigned 90% of each corpus as training data, and 10% as testing data.

4 Grammar Acquisition

Our grammar acquisition pipeline is shown in Figure 1. We use the following general-purpose tools and resources: the LT TTT tokenizer [Grover *et al.*, 2000], the Brill part-of-speech tagger [Brill, 1992], the Collins parser [Collins, 1999], WordNet [Fellbaum, 1998], and VerbNet [Kipper *et al.*, 2000]. Domain-specific replacements for any of these tools or resources can be made at any time. For example, in some of our experiments, we added the ontology and lexicon created for the TRIPS system [Dzikovska *et al.*, 2003].

Input text is automatically regularized, split into sentences and tokenized. Sentences containing quotations are removed. Then, the text is part-of-speech tagged and parsed, giving a set of parse trees, some with errors. We retain only the most likely parse for each sentence. Sentences that cannot be parsed are dropped from further processing.

A set of transformations are applied to the parse trees to convert all sentences to active, declarative form with one main verb only (clauses joined by a coordinating or subordinating conjunction are split).

We then try to identify the semantic roles in the sentence. Considerable research has been done on semantic role labeling (e.g. [Gildea and Juraksky, 2002; Pradhan *et al.*, 2004]); for this initial work, we simply look up the main verb of the sentence in VerbNet [Kipper *et al.*, 2000] and the hypernyms of the verb's arguments in WordNet [Fellbaum, 1998]. If the restrictions of the roles in a verb frame for the verb match the hypernyms of the verb's arguments in the sentence, we add semantic features to the parse tree, annotating the main verb with the verb frame and the verb's arguments with the semantic roles and hypernyms. A sentence can lead to multiple annotated trees if multiple verb frames match. Figure 2 shows a sample sentence tree extracted from the sentence "The cat went to the garden".

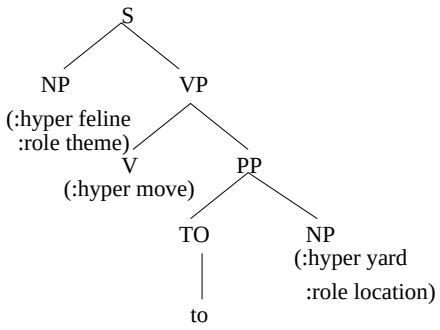

Figure 2: An initial tree for "the cat went to the garden"

Finally, we extract initial and auxiliary trees from the annotated parse trees. Initial tree types include S, NP, VP and PP; auxiliary tree types include NP, VP, PP, ADJP, ADVP and QP. We cluster these trees by their semantics and assign relative frequencies to each tree. Table 3 shows the number of initital trees we collected for each corpus. The grammars we obtain are probabilistic lexicalized tree-adjoining grammars with semantic features [Abeille and Rambow, 2000].

Table 2 shows the number of failures in our grammar acquisition process after parsing, after verb lookup and after verb frame matching. Note that the fact that the parser found a parse for 100% of the sentences in our corpora does not mean that it found a correct parse. We examined the 330 sentences from our CALO corpus for which there were no verb frames, and labeled them by hand as primarily spelling/tokenization (25), fragmentary (150), conventional (33), containing idiomatic language (16), containing highly unsyntactic language use (70), containing named entities that were problematic (15), or "other" (39). (The total sums to more than 330 because some sentences exhibited multiple phenomena.) We concluded that for this dialog corpus, some sentences just should not be included in the generation grammar. A similar examination of our EXPL corpus (469 'failure' sentences) gave 4 spelling/tokenization, 194 fragmentary, 6 idiomatic, 36 highly unsyntactic, 13 problematic named entities, and 230 other irregularities (mostly unusual verbs or unusual subject/verb combinations like 'react', 'typify' and 'integrate'). In this corpus, most of the fragments are titles or advertising.

The steep drop in data retention that results from an inability to find a matching verb frame is sometimes due to the lack of a suitable verb frame in our knowledge resources, sometimes due to the fact that the verb in question is a stative and sometimes due to parsing errors leading to misidentification of the verb. However, the majority of these errors are due to mismatches between the WordNet hypernyms of words filling sentential roles, and the semantic role type information in VerbNet. We were initially surprised to see such a steep rise in performance for our non-CALO corpora when the TRIPS lexicon and ontology are added, but with this lexicon and ontology these mismatches are not present. This type of mismatch is one of the largest problems with the use of general-purpose knowledge resources created by different researchers

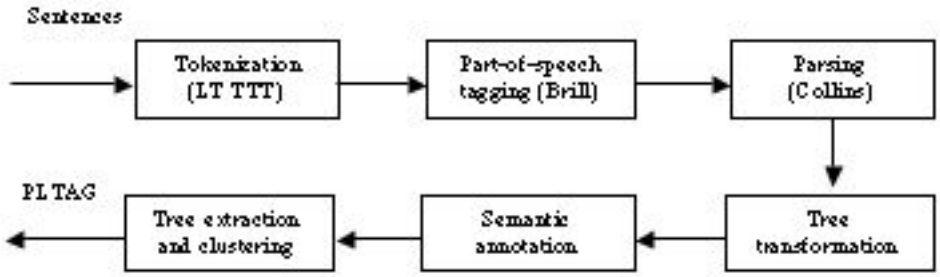

Figure 1: Architecture of grammar acquisition process'

Corpus	% parsed	% has verb frames	% into grammar
Specialized			
EXPL	100.00%	88.88%	53.02%
CALO	100.00%	89.80%	62.61%
WSJ	100.00%	64.35%	42.32%
NoParse	100.00%	71.87%	50.91%
General			
EXPL	100.00%	57.22%	41.95%
CALO	100.00%	57.24%	43.87%
WSJ	100.00%	40.06%	30.99%
NoParse	100.00%	48.94%	37.32%

Table 2: Grammar acquisition: Sources of data loss

for different purposes. In the future, we will use statistical semantic role labeling as described by [Gildea and Juraksky, 2002] to overcome the problem of type mismatches, and the smaller problem of missing verbs or missing verb frames.

5 Surface Realizer

We implemented a surface realizer for these grammars that uses a chart to efficiently compute all possible realizations of an input semantic form. The input is a logical form. Content words may be specified in the input; any lexical selection that needs to be done during surface realization is done using WordNet. We find all the initial or auxiliary trees covering each part of the input semantics and place them in the chart. We then fill in the chart with possible combinations of these trees using adjunction and substitution. Finally, we read off those trees that cover all or most of the input logical form and are rooted at S.

We use the same transformation rules used during grammar extraction to transform the resulting syntax tree if the input semantics specify a non-declarative sentence type. These transformation rules, and (if used for dialog) a small number of templates covering conventional utterances (greetings, thanks, acknowledgments) are the only hand-written rules used in any of our surface realizers [Stent, 1999]; for the experiments reported below, we excluded templates covering conventional forms from the surface realizer.

Our surface realizer has a second stage that assigns morphological inflection. We can rank sentences directly using the relative frequencies assigned to trees in the acquired grammars. Using an independence assumption, we simply multiply the probabilities of all the trees that go into producing the sentence. However, this method of computing the probability of a sentence ignores word likelihoods. If the surface realizer is doing lexical selection using a lexicon that is not domain-specific, the use of a third-stage language model can also be used to rerank sentences.

6 Experiments

We performed three sets of experiments. Our first experiment (General) was designed to look at performance when using only general-purpose tools and resources. We used only VerbNet and WordNet as our knowledge resources and we ignored the parse trees in the Penn Treebank, instead re-parsing those sentences. Our second experiment (NoParse) was designed to look at performance gains when using a treebank with general-purpose resources. We used the Penn Treebank only for this experiment. In our third experiment (Specialized), we added to our knowledge resources a specialized lexicon and ontology from the TRIPS dialog system [Dzikovska *et al.*, 2003]. For each set of experiments we report results when running our surface realizer both with and without a second-stage language model. For these experiments, we used a trigram language model trained on the Brown corpus to avoid artificially boosting performance on the Wall Street Journal with a Wall Street Journal-trained language model.

We used the same test input for each experiment. We constructed test input semi-automatically from the test data from each corpus. We used the same procedure used to acquire

Corpus	# unique initial trees	# np	# vp	#pp	# adjp	# advp	# qp
CALO	3206	133	57	1643	28	18	12
WSJ	25024	175	199	17705	95	63	47
EXPL	6345	101	89	4224	35	21	24

Table 3: Number of initial trees and auxiliary trees extracted from each corpus

Corpus	General	Specialized
EXPL	80.60%	89.69%
CALO	78.61%	88.72%
WSJ	77.39%	86.67%
NoParse	75.41%	86.97%
CALO + WSJ	78.61%	88.89%

Table 4: Coverage of generation grammars (number of sentences generated over number of inputs to generators)

our grammar (tokenize, part-of-speech tag, parse, annotate with semantic information), and then stripped off the logical form from the resulting tree. We retained content words (stemmed), but permitted the surface realizer to perform lexical selection of function words.

For each experiment, we evaluated using automatic and human evaluation. For comparison with previous work ([Callaway, 2003]), we report simple string accuracy (SSA) [Bangalore *et al.*, 2000]. We also report performance using Melamed's F measure [Turian *et al.*, 2003], which we have found to be somewhat more highly correlated with human judgments than the BLEU score [Stent *et al.*, 2005]. However, neither of these metrics works very well in the absence of multiple reference sentences. Therefore, we also conducted a human evaluation, for which we used the approach described in [Stent *et al.*, 2005].

6.1 Performance and Coverage

Our system was competitive in terms of performance. The average time taken to generate a single sentence for the one stage generation system is 48.41 milliseconds, while the average time for the two stage generation system is 58.72 milliseconds (cf. [Callaway, 2003]). Increasing the size of the grammar does slow this down somewhat, but we use very efficient storage of initial and auxiliary trees and can also resort to chart pruning if the chart becomes too large.

Data losses during grammar acquisition are reported earlier in this paper. These same loss rates apply to our testing data. We show the coverage of our acquired generation grammars, excluding these loss rates, in Table 4. These coverage results again reflect the potential gains from using targeted or specialized lexicons and ontologies.

6.2 Automatic evaluation

We compare the generated sentences with their referrence sentences using SSA and Melamed's F-measure. We show these results in Table 5 for the General experiment, and in Table 6 for the Specialized experiment.

This performance is significantly lower than that of HALOGEN run with a similar setup (.81) and SURGE (.89) [Callaway, 2003; Langkilde, 2002]; however, most errors were due to gaps and inconsistencies in our lexicons and vocabulary gaps between testing and training data.

Corpus	1 stage	2 stages
EXPL	.47 / .55	.52 / .59
CALO	.48 / .60	.52 / .63
WSJ	.45 / .55	.50 / .58
NoParse	.45 / .49	.49 / .52
CALO + WSJ	.48 / .60	.52 / .62

Table 5: General: SSA / Melamed's F for different generation grammars

Corpus	1 stage	2 stages
EXPL	.73 / .76	.75 / .77
CALO	.74 / .80	.76 / .81
WSJ	.71 / .76	.72 / .77
NoParse	.68 / .72	.70 / .73
CALO +WSJ	.74 / . 80	.76 / .80

Table 6: Specialized: SSA / Melamed's F for different generation grammars

Overall, the generator could not find a matching tree in the grammar for 10.3% of constituents in the testing sentences. We looked more closely at the CALO+WSJ case; without the WSJ, 9.6% of constituents in the CALO data did not have a matching tree in the grammar, while with it, 9.3% did not. We conclude that the use of this extra data does not help; as long as the training data covers most phenomena that might occur, increases in data size will only help refine the relative frequencies on trees in the grammar.

6.3 Human Evaluation

We selected 156 sentences from our testing data at random. We ran them through our one-stage and two-stage generators, with and without the TRIPS lexicon and ontology, obtaining four output sentences for each testing sentence. An evaluator rated the sentences for fluency and adequacy using the method described in [Stent *et al.*, 2005]; the evaluator was blind to the source of the sentence. The evaluator also marked several types of error that could occur in each sentence: missing verb, incorrect modal verb, absent negation, incorrect preposition, morphology error. Finally, the evaluator also rated the reference sentence for fluency only.

Tables 7 and 8 show the results of our human evaluation. The average score for fluency for the reference sentences was 4.6 (18 had a fluency score of 1). For the candidate sentences, the overall average score for accuracy was 2.5 and for fluency was 2.3 (61 had 5 for both scores; 262 had 1 for both scores).

Corpus	1 stage	2 stages
EXPL	1.6 / 1.7	1.8 / 1.8
CALO	3.1 / 3.2	3.1 / 3.4
WSJ	1.3 / 1.4	1.3 / 1.4
NoParse	1.1 / 1.2	1.4 / 1.5
CALO + WSJ	1.5 / 1.6	1.6 / 1.9

Table 7: General: Average fluency and accuracy for different generation grammars

Corpus	1 stage	2 stages
EXPL	3.4 / 4	3.4 / 4
CALO	3 / 3.4	3 / 3.4
WSJ	3.0 / 3.4	3.3 / 3.8
NoParse	2.5 / 2.6	2.9 / 2.9
CALO +WSJ	1.5 / 1.3	1.5 / 1.4

Table 8: Specialized: Average fluency and accuracy for different generation grammars

Where the reference sentence was length 20 or more (28 sentences), the average candidate sentence scores were 2.1 for adequacy and 1.9 for fluency. Where the reference sentence had length 19 or less (128 sentences), the average candidate sentence scores were 2.5 for adequacy and 2.3 for fluency.

There were 87 instances of a missing verb, 43 of an incorrect preposition, 8 of a missing negation, 36 of an incorrect modal (usually 'can' being replaced by 'will'), and 24 morphology errors (some duplicates). Other issues noticed by the evaluator included dropped passive forms, questions written as declarations, missing phrasal modifiers and relative clauses, and a few reference sentences that were nearquotations. However, because the test input was constructed using our imperfect generator acquisition pipeline, it is not clear for some of these errors (missing negation, incorrect modals, missing verbs, missing modifiers) whether the semantic content was included in the input logical form.

We conclude from this evaluation that in addition to improving our generator acquisition methodology, we should work on our transformation rules and on negation.

6.4 Discussion

We were surprised to find no performance improvement from using treebanked data. We think this result can be explained by the fact that these sentences were longer than those in the other corpus. We noticed during human evaluation that in long sentences, there were phrases that were coherent, but these phrases were arranged in incoherent ways.

We were also surprised to find no performance gains from adding WSJ data to our CALO surface realizer. We had hoped that this additional data would take care of phenomena unseen in the much smaller CALO corpus. Looking at our human evaluation, we now think that it served indeed to augment the number of trees, but not to augment the number of helpful trees.

From both the automatic and the human evaluation, we can see that there are substantial performance gains when using a grammar acquired with targeted lexicons and ontologies. Also, there are performance gains from using a 2-stage surface realizer, although the size of these gains decreases dramatically in the Specialized case. We think that with some more effort, we can acquire domain-specific grammars whose performance is not significantly different without a second stage sentence ranker.

7 Conclusions and Future Work

In this paper, we evaluated the possibility of using general-purpose tools and resources to acquire probabilistic grammars for generation from unannotated data. We found that mismatches between semantic category labels restricts the utility of general-purpose resources like WordNet and VerbNet, but that when a little domain-specific information is added performance improves dramatically.

In future work, we plan to improve our pipeline by the addition of statistical semantic role labeling and improved parsing. We will also correct logical form representation failings highlighted by our human evaluation. We are currently exploring knowledge-free approaches to this same task, including a classifier-based approach and a graph-based approach, that will also let us meet our goals of: high-quality domain-specific surface realization without treebanks; modeling of genre- and domain-specific variation; and on-line adaptation in dialog.

References

[Abeille and Rambow, 2000] Anne Abeille and Owen Rambow. *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*. Center for the Study of Language and Information, Stanford, California, 2000.

[Bangalore *et al.*, 2000] S. Bangalore, O. Rambow, and S. Whittaker. Evaluation metrics for generation. In *Proceedings of INLG 2000*, 2000.

[Baptist and Seneff, 2000] Lauren Baptist and Stephanie Seneff. Genesis-II: A Versatile System for Language Generation in Conversational System Applications. In *ICSLP-2000*, volume 3, pages 271–274, Beijing, China, 2000.

[Brill, 1992] E. Brill. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP 1992*, pages 152–155, Trento, IT, 1992.

[Callaway, 2003] C. Callaway. Evaluating coverage for large symbolic NLG grammars. In *Proceedings of IJCAI 2003*, 2003.

[Channarukul, 1999] Songsak Channarukul. Yag: A Template-Based Natural Language Generator For Real-Time Systems. Master's thesis, University of Wisconsin at Milwakee, 1999.

[Collins, 1999] M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[Creswell, 2003] C. Creswell. *Syntactic form and discourse function in natural language generation.* PhD thesis, University of Pennsylvania, 2003.

[Dzikovska *et al.*, 2003] M. Dzikovska, J. Allen, and M. Swift. Integrating linguistic and domain knowledge for spoken dialogue systems in multiple domains. In *Proceedings of the Workshop on Knowledge and Reasoning in Practical Dialog Systems, IJCAI 2003*, 2003.

[Elhadad and Robin, 1997] M. Elhadad and J. Robin. SURGE: a Comprehensive Plug-in Syntactic Realization Component for Text Generation, July 1997.

[Elhadad, 1993] M. Elhadad. FUF: the Universal Unifier User Manual v5.2. Technical report, Department of Computer Science, Ben Gurion University of the Negev, 1993.

[Fellbaum, 1998] Christiane Fellbaum. *WordNet, an Electronic Lexical Database*. MIT Press, 1998.

[Gildea and Juraksky, 2002] D. Gildea and D. Juraksky. Automatic labeling of semantic roles. *Computational Linguistics*, 2002.

[Grover *et al.*, 2000] C. Grover, C. Matheson, A. Mikheev, and M. Moens. LT TTT – a flexible tokenization tool. In *Proceedings of LREC 2000*, 2000.

[howstuffworks, 2003] How Stuff Works, 2003.

[Kipper *et al.*, 2000] K. Kipper, H. Dang, and M. Palmer. Class-Based Construction of a Verb Lexicon. In *AAAI-2000 Seventeenth National Conference on Artificial Intelligence*, Austin, TX, 2000.

[Langkilde, 2002] I. Langkilde. *A Foundation for General-Purpose Natural Language Generation: Sentence Realization using Probabilistic Models of Language*. PhD thesis, Information Science Institute, USC school of Engeenring, 2002.

[Marciniak and Strube, 2004] T. Marciniak and M. Strube. Classification-based generation using TAG. In *Proceedings of INLG 2004*, 2004.

[Marcus *et al.*, 1993] M. Marcus, M. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2), 1993.

[Oh and Rudnicky, 2002] A. Oh and A. Rudnicky. Stochastic natural language generation for spoken dialog systems. *Computer Speech and Language*, 16, October 2002.

[Pan and Shaw, 2004] S. Pan and J. Shaw. SEGUE: A hybrid case-based surface natural language generator. In *Proceedings of INLG 2004*, 2004.

[Pradhan *et al.*, 2004] S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. Shallow semantic parsing support vector machines. In *Proceedings of of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL-2004)*, Boston, USA, May 2-7 2004.

[Purver and Kempson, 2004] M. Purver and R. Kempson. Incremental context-based generation for dialogue. In *Proceedings of INLG 2004*, 2004.

[Rambow and Bangalore, 2000] O. Rambow and S. Bangalore. Using TAGs, a Tree Model, and a Language Model for Generation. In *Fifth International Workshop on Tree-Adjoining Grammars (TAG+)*, Paris, France, 2000. TAG+5.

[Ratnaparkhi, 2000] Adwait Ratnaparkhi. Trainable methods for surface natural language generation. In *the 1st Meeting of the North American Chapter of the Association of Computational Linguistics*, pages 194–201, Seattle, WA, 2000.

[S. Corston-Oliver and Moore, 2002] E. Ringger S. Corston-Oliver, M. Gamon and R. Moore. An overview of Amalgam: A machine-learned generation module. In *Proceedings of INLG 2002*, 2002.

[Stent *et al.*, 2005] A. Stent, M. Marge, and M. Singhai. Evaluating evaluation methods for generation in the presence of variation. In *Proceedings of CICLing 2005*, 2005.

[Stent, 1999] A. Stent. Content planning in continuous-speech spoken dialog systems. In *Proceedings of the KI'99 workshop "May I Speak Freely?"*, 1999.

[Turian *et al.*, 2003] J. Turian, L. Shen, and I. D. Melamed. Evalaution of machine translation and its evaluation. In *Proceedings of MT Summit IX*, 2003.

Concurrent Constraint Programming as a Whiteboard Architecture for Probabilistic NLG

Irene Langkilde-Geary
Computer Science Department
Brigham Young University
Provo, UT
irenelg@cs.byu.edu

1 Abstract

In a recent attempt to generalize across the architectures of 19 representative generator systems, Cahill et al. proposed a whiteboard as a reference design [Cahill *et al.*, 1999; 2000]. They argued that a whiteboard was general enough to subsume the variety of architectures surveyed and could serve as a standard to facilite the sharing and reuse of processing resources. A whiteboard architecture divides processing into knowledge-centric functional modules and models communication between modules with a single repository of data. The data stored in a whiteboard is added cumulatively by extending or copying existing data, but not changing or removing anything. A whiteboard thus generalizes further what had previously been considered the most general architecture for generation, namely a blackboard architecture, in which changes to the data repository are destructive, not cumulative.

Existing large-scale practical systems to-date have actually used other architectures (typically a pipeline) for the sake of simplicity. However, one ultimate goal in generation is to design a general-purpose system that is not only large-scale, but that works equally well across a variety of applications from dialog to machine translation to automatic summarization. A whiteboard architecture is necessary to achieve such generality, since different applications will provide somewhat different kinds of information as input, and will thus need to perform subtasks in different orders.

Historically there has been much discussion and puzzlement as to what sort of representation ought to be used as the input to a generator. For example, there has been uncertainty about what level of abstraction would be appropriate — deep (semantic) or shallow (syntactic). Deep inputs allow clients such as dialog systems to delegate much more work to a generator. However, deep (or even shallow) representations may be difficult for client applications like machine translation or summarization to construct.

Furthermore, the literature has noted that many real-world applications would find it convenient to generate output using a mix of canned text and templates in addition to fine-grained representations. It would therefore be desirable for a general-purpose generator to support all three forms of granularity.

Finally, the degree to which an input may be underspecified can vary. In general, client applications are likely to prefer having to provide the littlest amount of information necessary to get reasonable output, because that makes using the generator much easier. Sometimes the information needed to specify the input is simply unavailable. For example, in machine translation from Japanese to English, the generator needs to know the number feature for nouns, but that is not typically explicit in Japanese. The generator must then make its best guess, or choose a default. On the other hand, there are times when a client may want precise control over some or all of the output generated, for example to maintain coherency across multiple sentences or utterances.

Can a single system possibly do all this at once? I suggest that it can by allowing inputs to vary along the three dimensions of abstraction level, granularity, and degree of specification independently. Using a declarative (not transformational) representation of natural language sentences, this task can be modeled as a constraint satisfaction/optimization problem where each word is associated with a set of features, and some subset of the features have values given in the input while the remainder must be solved for. The framework is neutral with respect to which features are known initially versus unknown. The task then is to solve for the unknown values given the known ones using a set of constraints on the values of the features. The constraints may include rules and/or probabilistic dependencies or any other declarative specification.

One of the main features in this model is the identity of each word's syntactic head, and another is the position of a word in the syntax tree with respect to its head and siblings (assuming a dependency-style representation). A third feature is the linear position in the final sequence of words. Assuming projectivity, the tree position and parent ID features are sufficient to determine the linear order of words of a realized sentence given an input. Conversely, there is no reason why one cannot solve for the parent ID and tree positions of each word in a sentence given the linear positions.

Thus, such an approach to generation could also do parsing. Handling inputs with different granularities then is simply a matter of parsing canned text or template fragments and integrating the resulting fine-grained specifications with other fine-grained specifications to produce an output.

The recent maturation of concurrent constraint programming (CCP) makes such a formulation possible. Constraint programming is a flexible, general, and principled framework for efficiently solving hard combinatorial problems. It integrates techniques developed in AI and Operations Research

and embeds them within a general-purpose programming language. Concurrent constraint programs are the most powerful form of constraint programs, because they increase the modularity and declarativeness with which a program can be formulated. In particular, they offer a greater degree of order-independence than non-concurrent formulations. A concurrent constraint program's behavior is implicitly analogous to that of a whiteboard architecture (although the specific representation and processing details may differ from those proposed by [Cahill *et al.*, 2002]).

In this talk I will present my lab's work on formulating probabilistic parsing/generation as a CCP. This early stage work helps to demonstrate the feasibility and effectiveness of the approach.

References

[Cahill *et al.*, 1999] L. Cahill, C. Doran, R. Evans, C. Mellish, D. Paiva, M. Reape, D. Scott, and N. Tipper. In search of a reference architecture for NLG systems. In *Proc. EWNLG*, 1999.

[Cahill *et al.*, 2000] L. Cahill, C. Doran, R. Evans, R. Kibble, C. Mellish, D. Paiva, M. Reape, D. Scott, and N. Tipper. Enabling resource sharing in language generation: an abstract reference architecture. In *In Proc. LREC*, 2000.

[Cahill *et al.*, 2002] L. Cahill, R. Evans, C. Mellish, D. Paiva, M. Reape, and D. Scott. The RAGS reference manual. Technical report, Univ. of Brighton and Univ. of Edinburgh, 2002.