Evaluating automatic extraction of rules for sentence plan construction

Amanda Stent
AT&T Labs – Research
Florham Park, NJ, USA
stent@research.att.com

Martin Molina
Department of Artificial Intelligence
Universidad Politécnica de Madrid, Spain
martin.molina@upm.es

Abstract

The freely available SPaRKy sentence planner uses hand-written weighted rules for sentence plan construction, and a user- or domain-specific second-stage ranker for sentence plan selection. However, coming up with sentence plan construction rules for a new domain can be difficult. In this paper, we automatically extract sentence plan construction rules from the RST-DT corpus. In our rules, we use only domain-independent features that are available to a sentence planner at runtime. We evaluate these rules, and outline ways in which they can be used for sentence planning. We have integrated them into a revised version of SPaRKy.

1 Introduction

Most natural language generation (NLG) systems have a pipeline architecture consisting of four core stages: content selection, discourse planning, sentence planning, and surface realization (Reiter and Dale, 2000; Rambow et al., 2001). A sentence planner maps from an input discourse plan to an output sentence plan. As part of this process it performs several tasks, including sentence ordering, sentence aggregation, discourse cue insertion and perhaps referring expression generation (Stent et al., 2004; Walker et al., 2007; Williams and Reiter, 2003).

The developer of a sentence planner must typically write rules by hand (e.g. (Stent et al., 2004; Walker et al., 2007)) or learn a domain-specific model from a corpus of training data (e.g. (Williams and Reiter, 2003)). Unfortunately, there are very few corpora annotated with discourse plans, and it is hard to automatically label a corpus for discourse structure. It is also hard to hand-write sentence planning rules starting from a "blank slate", as it were.

In this paper, we outline a method for extracting sentence plan construction rules from the only publicly available corpus of discourse trees, the RST Discourse Treebank (RST-DT) (Carlson et al., 2002). These rules use only domain-independent information available to a sentence planner at run-time. They have been integrated into the freely-available SPaRKy sentence planner. They serve as a starting point for a user of SPaRKy, who can add, remove or modify rules to fit a particular domain.

We also describe a set of experiments in which we look at each sentence plan construction task in order, evaluating our rules for that task in terms of coverage and discriminative power. We discuss the implications of these experiments for sentence planning.

The rest of this paper is structured as follows: In Section 2 we describe the sentence planning process using SPaRKy as an example. In Sections 3 through 5 we describe how we obtain sentence plan construction rules. In Section 6, we evaluate alternative rule sets. In Section 7, we describe our modifications to the SPaRKy sentence planner to use these rules. In Section 8 we conclude and present future work.

2 Sentence Planning in SPaRKy

The only publicly available sentence planner for data-to-text generation is SPaRKy (Stent et al., 2004). SPaRKy takes as input a discourse plan (a tree with rhetorical relations on the internal nodes and a proposition representing a text *span* on each leaf), and outputs one or more sentence plans

Proceedings of SIGDIAL 2009: the 10th Annual Meeting of the Special Interest Group in Discourse and Dialogue, pages 290–297,
Queen Mary University of London, September 2009.

(each a tree with discourse cues and/or punctuation on the internal nodes). SPaRKy is a two-stage sentence planner. First, possible sentence plans are *constructed* through a sequence of decisions made using only local information about single nodes in the discourse plan. Second, the possible sentence plans are *ranked* using a user- or domain-specific sentence plan ranker that evaluates the global quality of each sentence plan (Walker et al., 2007).

Sentence plan construction in SPaRKy involves three tasks: *span ordering*, *sentence aggregation* (deciding whether to realize a pair of propositions as a single clause, a single sentence, or two sentences), and *discourse cue selection*[1]. SPaRKy uses a single set of hand-written weighted rules to perform these tasks. In the current distributed version of SPaRKy, there are 20 rules covering 9 discourse cues (*and*, *because*, *but*, *however*, *on the other hand*, *since*, *while*, *with*, and the default, *period*). Each rule operates on the children of one rhetorical relation, and may impose an ordering, insert punctuation or merge two propositions, and/or insert a discourse cue. During sentence plan construction, SPaRKy walks over the input discourse plan, at each node finding all matching rules and applying one which it selects probabilistically according to the rule weights (with some randomness to permit variation).

While the developer of a NLG system will always have to adapt the sentence planner to his or her domain, it is often hard to come up with sentence planning rules "from scratch". As a result of the work described here a SPaRKy user will have a solid foundation for sentence plan construction.

3 Data

We use the Wall Street Journal Penn Treebank corpus (Marcus et al., 1993), which is a corpus of text annotated for syntactic structure. We also use two additional annotations done on (parts of) that corpus: PropBank (Kingsbury and Palmer, 2003), which consists of annotations for predicate-argument structure; and the RST-DT (Carlson et al., 2002), which consists of annotations for rhetorical structure.

We had to process this data into a form suitable for feature extraction. First, we produced a flattened form of the syntactic annotations, in which each word was labeled with its part-of-speech tag and the path to the root of the parse tree. Each word was also assigned indices in the sentence (so we could apply the PropBank annotations) and in the document (so we could apply the RST-DT annotations)[2].

Second, we attach to each word one or more labels from the PropBank annotations (each label consists of a predicate index, and either a predicate name or a semantic role type and index).

Third, we extract relation information from the RST-DT. For each relation, we extract the relation name, the types of each child ("Nucleus" or "Satellite"), and the start and end word indices for each child. Finally, we extract from the word-level annotations the marked-up words for each text span in each rhetorical relation.

4 Features

Features are individual rule conditions. In the standard NLG pipeline, no information about the realized text is available to the sentence planner. However, existing sentence planners use lexical and word sequence information to improve performance for a particular domain. Williams and Reiter (2003) appear to do surface realization before sentence planning, while Walker et al. (2007) perform surface realization between sentence plan construction and sentence plan ranking. We are concerned with sentence plan construction only; also, we want to produce sentence plan construction rules that are as domain-independent as possible. So we use no features that rely on having realized text. However, we assume that the input propositions have been fairly well fleshed-out, so that one has information about predicate-argument structure, tense, and the information status of entities to be realized.

A relation has a label as well as one or more child text *spans*. The features we extract from our data include both per-span and per-relation features. In our experiments we use a subset of these features which is fairly domain-independent and does not overly partition our data. The complete set of features (*full*) is as well as our *reduced* set are given in Table 1.

[1] SPaRKy also does some referring expression generation, in a single pass over each completed sentence plan.

[2] The Penn Treebank and the RST-DT segment words and punctuation slightly differently, which makes it hard to align the various annotations.

Feature type	*Full feature set*	*Reduced feature set*
Per-relation	relation, relation is leaf, parent relation, span coref, combined verb type class, combined verb type, identifier of shortest span, temporal order of spans	relation, relation is leaf, parent relation, span coref, combined verb type class, identifier of shortest span, temporal order of spans
Per-span, span identifier	span identifier	span identifier
Per-span, span length	number of NPs in span	
Per-span, span verb	verb type class, verb type, verb part of speech, verb is negated, verb has modal	
Per-span, arguments	argument status for ARG0 to ARG5 plus ARGM-{EXT, DIR, LOC, TMP, REC, PRD, ADV, MNR, CAU, PNC}	

Table 1: Features used in evaluation

4.1 Per-Span Features

We extract per-span features from *basic* spans (leaves of the RST tree) and from *complex* spans (internal nodes of the RST tree). For each span we compute: identifier, text, length, verb information, span argument information, discourse cue information, and span-final punctuation.

Identifier We need a way to refer to the child spans in the rules. For relations having only one child span of each type (Satellite or Nucleus), we order the spans by type. Otherwise, we order the spans alphabetically by span text. The span identifier for each child span is the index of the span in the resulting list.

Text We extract the text of the span, and the indices of its first and last words in the Penn Treebank. We only use this information during data extraction. However, in a system like that of Williams and Reiter (Williams and Reiter, 2003), where sentence planning is done after or with surface realization, these features could be used. They could also be used to train a sentence plan ranker for SPaRKy specific to the news domain.

Length We use the number of base NPs in the span (as we cannot rely on having the complete realization during sentence planning).

Verb We extract verb type, which can be *N/A* (there is no labeled predicate for the span), *stat* (the span's main verb is a form of "to be"), a single PropBank predicate (e.g. *create.01*), or *mixed* (the span contains more than one predicate). We then abstract to get the verb type class: *N/A*, *pb* (a PropBank predicate), *stat*, or *mixed*.

If the span contains a single predicate or multiple predicates all having the same part-of-speech tag, we extract that (as an indicator of tense). We also extract information about negation and modals (using the PropBank tags ARGM-NEG and ARGM-MOD).

Arguments We extract the text of the arguments of the predicate(s) in the span: ARG0 to ARG5, as well as ARGM-{EXT, DIR, LOC, TMP, REC, PRD, ADV, MNR, CAU, PNC}. We then abstract to get an approximation of information status. An argument status feature covers zero or more instantiations of the argument and can have the value *N/A* (no instantiations), *proper* (proper noun phrase(s)), *pro* (pronoun(s)), *def* (definite noun phrase(s)), *indef* (indefinite noun phrase(s)), *quant* (noun phrase (s) containing quantifiers), *other* (we cannot determine a value), or *mixed* (the argument instantiations are not all of the same type).

Discourse Cues We extract discourse cue information from basic spans and from the first basic span in complex spans. We identify discourse cue(s) appearing at the start of the span, inside the span, and at the end of the span. PropBank includes the argument label ARGM-DIS for discourse cues; however, we adopt a more expansive notion of discourse cue. We say that a discourse cue can be *either*: any sequence of words all labeled ARGM-DIS and belonging to the same predicate, anywhere in the span; *or* any cue from a (slightly expanded version of) the set of cues studied by Marcu (Marcu, 1997), if it appears at the start of a span, at the end of a span, or immediately before or after a comma, *and* if its lowest containing phrase tag is one of {ADJP, ADVP, CONJP, FRAG, NP-ADV, PP, UCP, SBAR, WH} *or* its part of speech tag is one of {CC, WDT}[3].

Punctuation We extract punctuation (*N/A* or . or ? or ! or ; or : or ,) at the end of the span.

[3]We constructed these rules by extracting from the WSJ Penn Treebank all instances of the cues in Marcu's list, and then examining instances where the word sequence was not actually a discourse cue. Some mistakes still occur in cue extraction.

4.2 Per-Relation Features

For each relation we compute: name, the combined verb type and verb class of the child spans, whether any argument instantiations in the child spans are coreferential, and which child span is shortest (or the temporal order of the child spans).
Relation, Parent Relation The core relation label for the relation and its parent relation (e.g. *attribution* for *attribution-e* and *attribution-n*).
Relation is Leaf True if child spans of the relation are leaf spans (not themselves relations).
Combined Verb The shared verb for the relation: the child spans' verb type if there is only one non-*N/A* verb type among the child spans; otherwise, *mixed*. We then abstract from the shared verb type to the shared verb type class.
Span Coreference We use the information PropBank gives about intra-sentential coreference. We do not employ any algorithm or annotation to identify inter-sentential coreference.
Shortest Span The identifier of the child span with the fewest base NPs.
Temporal Order of Spans For some relations (e.g. *sequence, temporal-before, temporal-after*), the temporal order is very important. For these relations we note the temporal order of the child spans rather than the shortest span.

5 Rule Extraction

Each rule we extract consists of a set of per-relation and per-span features (the *conditions*), and a *pattern* (the *effects*). The conditions contain either: the relation only, features from the reduced feature set, or features from the full feature set. The pattern can be an ordering of child spans, a set of between-span punctuation markers, a set of discourse cues, or an ordering of child spans mixed with punctuation markers and discourse cues. Each extracted rule is stored as XML.

We only extract rules for relations having two or more children. We also exclude RST-DT's *span* and *same-unit* relations because they are not important for our task. Finally, because the accuracy of low-level (just above the span) rhetorical relation annotation is greater than that of high-level relation annotation, we extract rules from two data sets: one only containing *first-level* relations (those whose children are all basic spans), and one containing *all* relations regardless of level in the RST tree. The output from the rule extraction process is six alternative rule sets for each sentence plan construction task: first-level or all data, with either the relation condition alone, the reduced feature set, or the full feature set.

Concession rule:
conditions:
type child="0": nucleus, type child="1": satellite, shortest: 0, isCoref: 0, isLeaf: 1, isSamePredClass: mixed, numChildren: 2, relation: concession, parentRel: antithesis
effects:
order: 1 0, punc child="1": comma, cues child="1": while
example:
(1) While some automotive programs have been delayed, (0) they have n't been canceled

Sequence rule:
conditions:
type child="0": nucleus, type child="1": nucleus, type child="2": nucleus, type child="3": nucleus, isCoref: 1, isLeaf: 1, isSamePredClass: mixed, numChildren: 4, relation: sequence, parentRel: circumstance, temporalOrder: 0 1 2 3
effects:
order: 0 1 2 3, punc child="0": comma, punc child="1": comma, punc child="2": n/a, cues child="3": and
example:
(0) when you can get pension fund money, (1) buy a portfolio, (2) sell off pieces off it (3) and play your own game

Purpose rule:
conditions:
type child="0": nucleus, type child="1": satellite, shortest: 0, isCoref: 0, isLeaf: 0, isSamePredClass: shared, numChildren: 2, relation: purpose, parentRel: list
effects:
order: 0 1, punc child="0": n/a, cues child="1": so
example:
(0) In a modern system the government 's role is to give the people as much choice as possible
(1) so they are capable of making a choice

Figure 1: Glosses of extracted sentence planning rules for three relations (reduced feature set)

The maximum number of patterns we could have is 7680 per relation, if we limit ourselves to condition sets, relation instances with only two child spans, and a maximum of one discourse cue to each span (two possible orderings for child spans * four possible choices for punctuation * 480 choices for discourse cue on each span). By contrast, for our *all* data set there are 5810 unique rules conditioned on the reduced feature set (109.6 per relation) and 292 conditioned on just the relation (5.5 per relation). Example rules are given in Figure 1. Even though the data constrains sentence planning choices considerably, we still have many rules (most differing only in discourse cues).

6 Rule Evaluation

6.1 On Evaluation

There are two basic approaches to NLG, *text-to-text generation* (in which a model learned from a text corpus is applied to produce new texts from text input) and *data-to-text generation* (in which non-text input is converted into text output). In text-to-text generation, there has been considerable work on sentence fusion and information ordering, which are partly sentence planning tasks. For evaluation, researchers typically compare automatically produced text to the original human-produced text, which is assumed to be "correct" (e.g. (Karamanis, 2007; Barzilay and McKeown, 2005; Marsi and Krahmer, 2005)). However, an evaluation that considers the only "correct" answer for a sentence planning task to be the answer in the original text is overly harsh. First, although we assume that all the possibilities in the human-produced text are "reasonable", some may be awkward or incorrect for particular domains, while other less frequent ones in the newspaper domain may be more "correct" in another domain. Our purpose is to lay out sentence plan construction possibilities, not to reproduce the WSJ authorial voice. Second, because SPaRKy is a two-stage sentence planner and we are focusing here on sentence plan construction, we can only evaluate the local decisions made during that stage, not the overall quality of SPaRKy's output.

Evaluations of sentence planning tasks for data-to-text generation have tended to focus solely on discourse cues (e.g. (Eugenio et al., 1997; Grote and Stede, 1998; Moser and Moore, 1995; Nakatsu, 2008; Taboada, 2006)). By contrast, we want good coverage for all core sentence planning tasks. Although Walker *et al.* performed an evaluation of SPaRKy (Stent et al., 2004; Walker et al., 2007), they evaluated the output from the sentence planner as a whole, rather than evaluating each stage separately. Williams and Reiter, in the work most similar to ours, examined a subset of the RST-DT corpus to see if they could use it to perform span ordering, punctuation selection, and discourse cue selection and placement. However, they assumed that surface realization was already complete, so they used lexical features. Their sentence planner is not publicly available.

In the following sections, we evaluate the information in our sentence plan construction rules in terms of *coverage* and *discriminative power*. The first type of evaluation allows us to assess the degree to which our rules are general and provide system developers with an adequate number of choices for sentence planning. The second type of evaluation allows us to evaluate whether our reduced feature set helps us choose from the available possibilities better than a feature set consisting simply of the relation (i.e. is the complicated feature extraction necessary). Because we include the full feature set in this evaluation, it can also be seen as a text-to-text generation type of evaluation for readers who would like to use the sentence planning rules for news-style text generation.

6.2 Coverage

In our evaluation of coverage, we count the number of relations, discourse cues, and patterns we have obtained, and compare against other data sets described in the research literature.

6.2.1 Relation Coverage

There are 57 unique core relation labels in the RST-DT. We exclude *span* and *same-unit*. Two others, *elaboration-process-step* and *topic-comment*, never occur with two or more child spans. Our *first-level* and *all* rules cover all of the remaining 53. The most frequently occurring relations are *elaboration-additional*, *list*, *attribution*, *elaboration-object-attribute*, *contrast*, *circumstance* and *explanation-argumentative*.

By contrast, the current version of SPaRKy covers only 4 relations (*justify, contrast, sequence*, and *infer*)[4].

Mann and Thompson originally defined 24 relations (Mann and Thompson, 1987), while Hovy and Maier listed about 70 (Hovy and Maier, 1992).

6.2.2 Discourse Cue Coverage

Our *first-level* rules cover 92 discourse cues, and our *all* rules cover 205 discourse cues. The most commonly occurring discourse cues in both cases are *and*, *but*, *that*, *when*, *as*, *who* and *which*.

By contrast, the current version of SPaRKy covers only about 9 discourse cues.

In his dissertation Marcu identified about 478 discourse cues. We used a modified version of Marcu's cue list to extract discourse cues from our corpus, but some of Marcu's discourse cues do not occur in the RST-DT.

[4]Curiously, only two of these relations (*contrast* and *sequence*) appear in the RST-DT data (although *infer* may be equivalent to *span*).

6.2.3 Sentence Plan Pattern Coverage

For the *first-level* data we have 140 unique sentence plan patterns using the relation condition alone, and 1767 conditioning on the reduced feature set. For the *all* data we have 292 unique patterns with relation condition alone and 5810 with the reduced feature set. Most patterns differ only in choice of discourse cue(s).

No system developer will want to examine all 5810 rules. However, she or he may wish to look at the patterns for a particular relation. In our use of SPaRKy, for example, we have extended the patterns for the *sequence* relation by hand to cover temporal sequences of up to seven steps.

6.3 Discriminative Power

In this evaluation, we train decision tree classifiers for each sentence plan construction task. We experiment with both the *first-level* and *all* data sets and with both the *reduced* and *full* feature sets. For each experiment we perform ten-fold cross-validation using the J48 decision tree implementation provided in Weka (Witten and Eibe, 2005) with its default parameters. We also report performance for a model that selects a pattern conditioning only on the relation. Finally, we report performance of a baseline which always selects the most frequent pattern.

We evaluate using 1-best classification accuracy, by comparing with the choice made in the Penn Treebank for that task. We test for significant differences between methods using Cochran's Q, followed by post-hoc McNemar tests if significant differences existed. We also report the features with information gain greater than 0.1.

6.3.1 Span Ordering

We have one input feature vector for each relation instance that has two children[5]. In the feature vector, child spans are ordered by their identifiers, and the pattern is either *0_1* (first child, then second child) or *1_0* (second child, then first child).

Classification accuracy for all methods is reported in Table 2. All methods perform significantly better than baseline ($p < .001$), and both the reduced and full feature sets give results significantly better than using the relation alone ($p < .001$). The full feature set performs significantly better than the reduced feature set for the *all* data set ($p < .001$), but not for the *first-level* data set.

Most of the relations have a strong preference for one ordering or the other. Most mistakes are made on those that don't (e.g. *attribution, list*).

	First-level	All
Baseline	71.8144	71.4356
Per-relation	84.2707	82.3894
Reduced	89.6092	90.3147
Full	90.2129	91.9666

Table 2: Span ordering classification accuracy. For first-level data, n = 3147. For all data, n = 10170. Labels = {*0_1, 1_0*}.

	First-level	All
Baseline	74.5154	50.4425
Per-relation	74.5154	64.2773
Reduced	77.8201	72.1731
Full	74.3883	66.1357

Table 3: Between-span punctuation classification accuracy. For first-level data, n = 3147. For all data, n = 10170. Labels = {*semicolon, comma, full, N/A*}.

6.3.2 Punctuation Insertion

We have one input feature vector for each relation instance that has two children. We assume that span ordering is performed prior to punctuation insertion, so the child spans are ordered as they appear in the data. The pattern is the punctuation mark that should appear between the two child spans (one of *N/A* or *comma* or *semicolon* or *full*[6]), which indicates whether the two children should be realized as separate sentences, as separate clauses, or merged.

Classification accuracy for all methods is reported in Table 3. For the *all* data set, all methods perform significantly better than baseline ($p < .001$), and both the reduced and full feature sets give results significantly better than using the relation alone ($p < .001$). Furthermore, the reduced feature set performs significantly better than the full feature set ($p < .001$). By contrast, for the *first-level* data set, the reduced feature set performs significantly better than all the other data sets, while there are no statistically significant differences in performance between the baseline, per-relation and full feature sets.

The most common type of error was misclassifying *comma, semicolon* or *full* as *N/A*: for the

[5] The number of relation instances with three or more child spans is less than 2% of the data. Removing these relations made it feasible for us to train classifiers without crashing Weka.

[6] *full* indicates a sentence boundary (. or ? or !).

	First-level	All
Baseline	62.6629	68.4267
Per-relation	68.605	70.1377
Reduced	73.6257	73.9135
Full	74.3565	74.5919

Table 4: Discourse cue classification accuracy. For first-level data, n = 3147 and no. labels = 92. For all data, n = 10170 and no. labels = 203.

first-level data this is what the models trained on the per-relation and full feature sets do most of the time. The second most common type of error was misclassifying *comma, semicolon* or *N/A* as *full*.

6.3.3 Discourse cue selection

We have one input feature vector for each relation instance having two children. We use the same features as in the previous experiment, and as in the previous experiment, we order the child spans as they appear in the data. The pattern is the first discourse cue appearing in the ordered child spans[7].

Classification accuracy for all methods is reported in Table 4. All methods perform significantly better than baseline ($p < .001$), and both the reduced and full feature sets give results significantly better than using the relation alone ($p < .001$). The performance differences between the reduced and full feature sets are not statistically significant for either data set.

For this task, 44 of the 92 labels in the *first-level data*, and 97 of the 203 labels in the *all data*, occurred only once. These cues were typically mislabeled. Commonly occurring labels were typically labeled correctly.

6.4 Discussion

Our methods for rule extraction are not general in the sense that they rely on having access to particular types of annotation which are not widely available nor readily obtainable by automatic means. However, our extracted rules have quite broad coverage and will give NLG system developers a jump start when using and adapting SPaRKy.

Our reduced feature set compares favorably in discriminative power to both our full feature set and the per-relation feature set. It achieves a very good fit to the input data for the span ordering task and a good fit to the input data for the punctuation and discourse cue insertion tasks, especially for the *first-level* data set. Factors affect performance include: the punctuation insertion data is highly imbalanced (by far the most common label is *N/A*), while for the discourse cue insertion task there is a problem of data sparsity.

[7] Some relations have multiple cues, either independent cues such as *but* and *also*, or cues that depend on each other such as *on the one hand* and *on the other hand*. Using all cues is infeasible, and there are too few span-internal and span-final cues to break up the cue classification for this evaluation.

7 Revised SPaRKy

One way to use these results would be to model the sentence planning task as a cascade of classifiers, but this method does not permit the system developer to add his or her own rules. So we continue to use SPaRKy, which is rule-based. We have made several changes to the Java version of SPaRKy to support application of our sentence plan construction rules. We modified the classes for storing and managing rules to read our XML rule format and process rule conditions and patterns. We stripped out the dependence on RealPro and added hooks for SimpleNLG (Gatt and Reiter, 2009). We modified the rule application algorithm so that users can choose to use a single rule set with patterns covering all three sentence planning tasks, or one rule set for each sentence planning task. Also, since there are now many rules, we give the user the option to specify which relations jSPaRKy should load rules for at each run.

Information about the revised jSparky, including how to obtain it, is available at *http://www.research.att.com/~stent/sparky2.0/* or by contacting the first author.

8 Conclusions and Future Work

In this paper we described how we extracted less domain-dependent sentence plan construction rules from the RST-DT corpus. We presented evaluations of our extracted rule sets and described how we integrated them into the freely-available SPaRKy sentence planner.

In future work, we will experiment with discourse cue clustering. We are also looking at alternative ways of doing sentence planning that permit a tighter interleaving of sentence planning and surface realization for improved efficiency and output quality.

References

R. Barzilay and K. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

L. Carlson, D. Marcu, and M. E. Okurowski. 2002. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the SIGdial workshop on discourse and dialogue*.

B. Di Eugenio, J. D. Moore, and M. Paolucci. 1997. Learning features that predict cue usage. In *Proceedings of the EACL*.

A. Gatt and E. Reiter. 2009. SimpleNLG: A realisation engine for practical applications. In *Proceedings of the European Workshop on Natural Language Generation*.

B. Grote and M. Stede. 1998. Discourse marker choice in sentence planning. In *Proceedings of the 9th International Workshop on Natural Language Generation*.

E. Hovy and E. Maier. 1992. Parsimonious or profligate: how many and which discourse structure relations? Available from http://handle.dtic.mil/100.2/ADA278715.

N. Karamanis. 2007. Supplementing entity coherence with local rhetorical relations for information ordering. *Journal of Logic, Language and Information*, 16(4):445–464.

P. Kingsbury and M. Palmer. 2003. PropBank: the next level of TreeBank. In *Proceedings of the Workshop on Treebanks and Lexical Theories*.

B. Lavoi and O. Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of ANLP*.

W. Mann and S. Thompson. 1987. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, Information Sciences Institute, Los Angeles, CA.

D. Marcu. 1997. *The rhetorical parsing, summarization, and generation of natural language texts*. Ph.D. thesis, Department of Computer Science, University of Toronto.

M. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

E. Marsi and E. Krahmer. 2005. Explorations in sentence fusion. In *Proceedings of the European Workshop on Natural Language Generation*.

M. Moser and J. D. Moore. 1995. Using discourse analysis and automatic text generation to study discourse cue usage. In *Proceedings of the AAAI 1995 Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*.

C. Nakatsu. 2008. Learning contrastive connective in sentence realization ranking. In *Proceedings of SIGdial 2008*.

O. Rambow, S. Bangalore, and M. A. Walker. 2001. Natural language generation in dialog systems. In *Proceedings of HLT*.

E. Reiter and R. Dale. 2000. *Building natural language generation systems*. Cambridge University Press, Cambridge, UK.

A. Stent, R. Prasad, and M. A. Walker. 2004. Trainable sentence planning for complex information presentations in spoken dialog systems. In *Proceedings of the ACL*.

M. Taboada. 2006. Discourse markers as signals (or not) of rhetorical relations. *Journal of Pragmatics*, 38(4):567–592.

M. A. Walker, A. Stent, F. Mairesse, and R. Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, 30:413–456.

S. Williams and E. Reiter. 2003. A corpus analysis of discourse relations for natural language generation. In *Proceedings of Corpus Linguistics*.

I. Witten and F. Eibe. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition.